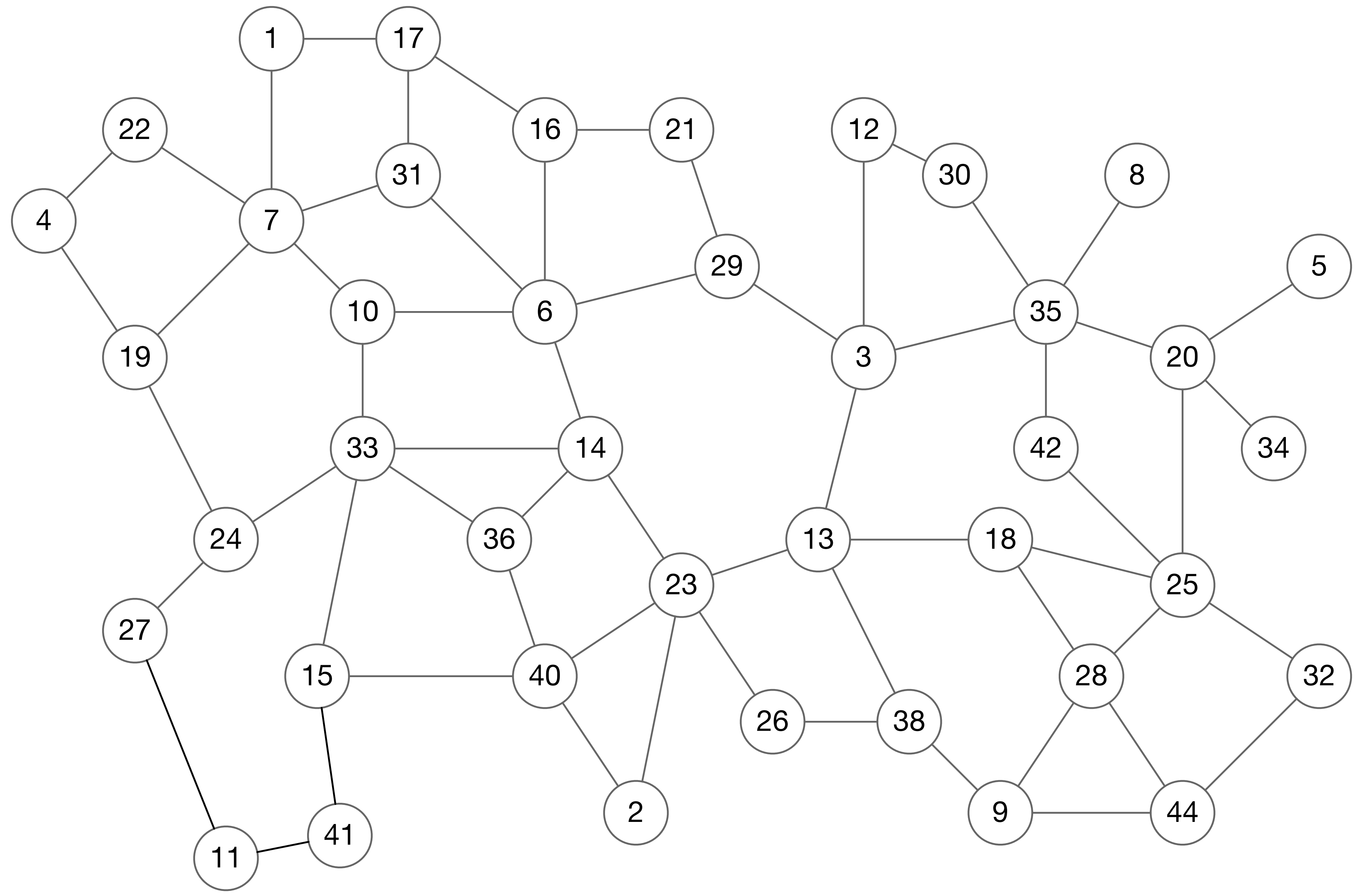
Distributed Edge Coloring in Time Polylogarithmic in Δ

Dennis Olivetti
Gran Sasso Science Institute, L'Aquila, Italy

Joint work with: Alkida Balliu, Sebastian Brandt, Fabian Kuhn

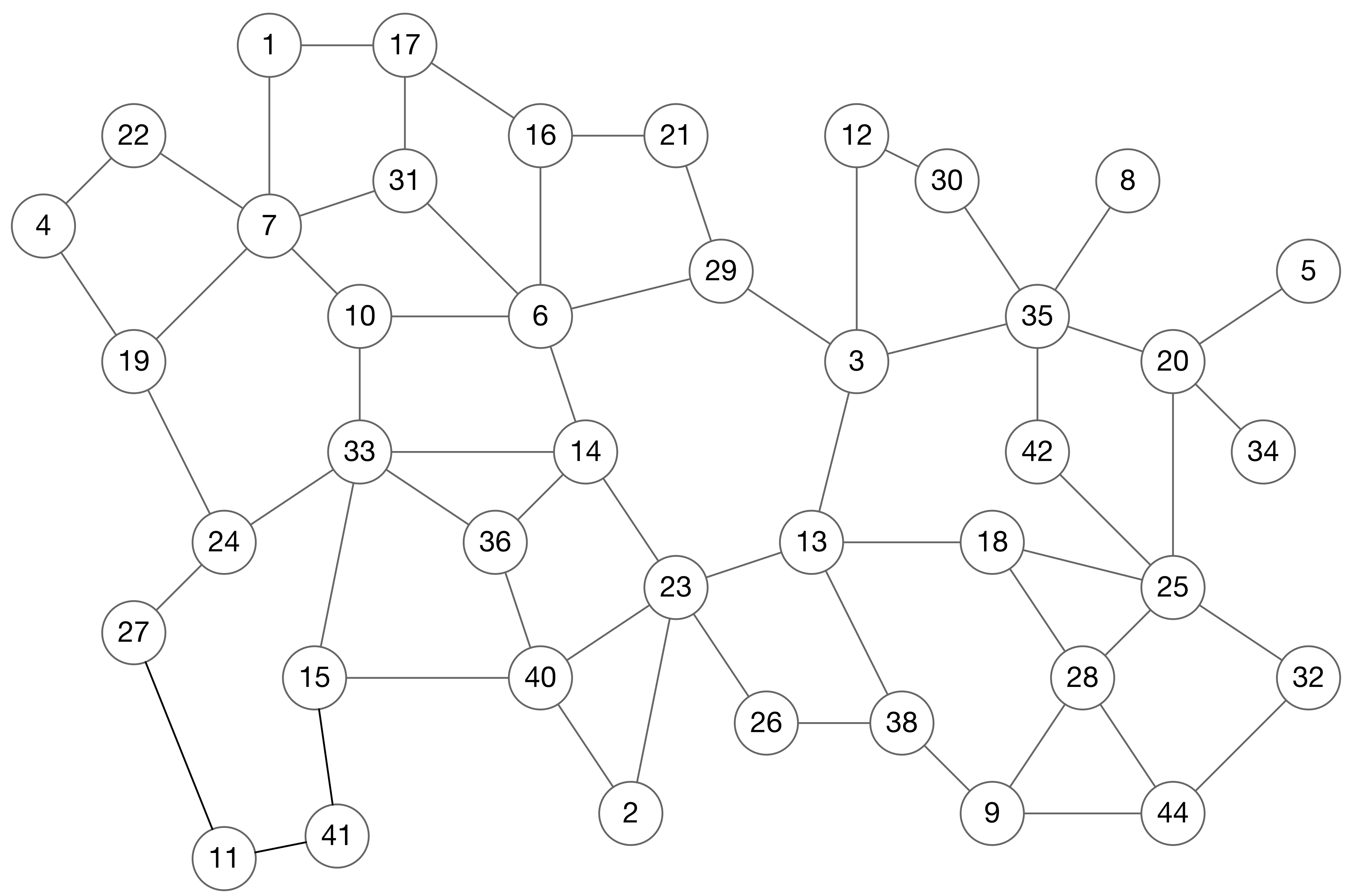
LOCAL model

- Undirected simple graph $G = (V, E)$ of n nodes and maximum degree Δ
- Each node has a unique ID
- Synchronous message passing model
- Unbounded computation
- Unbounded bandwidth

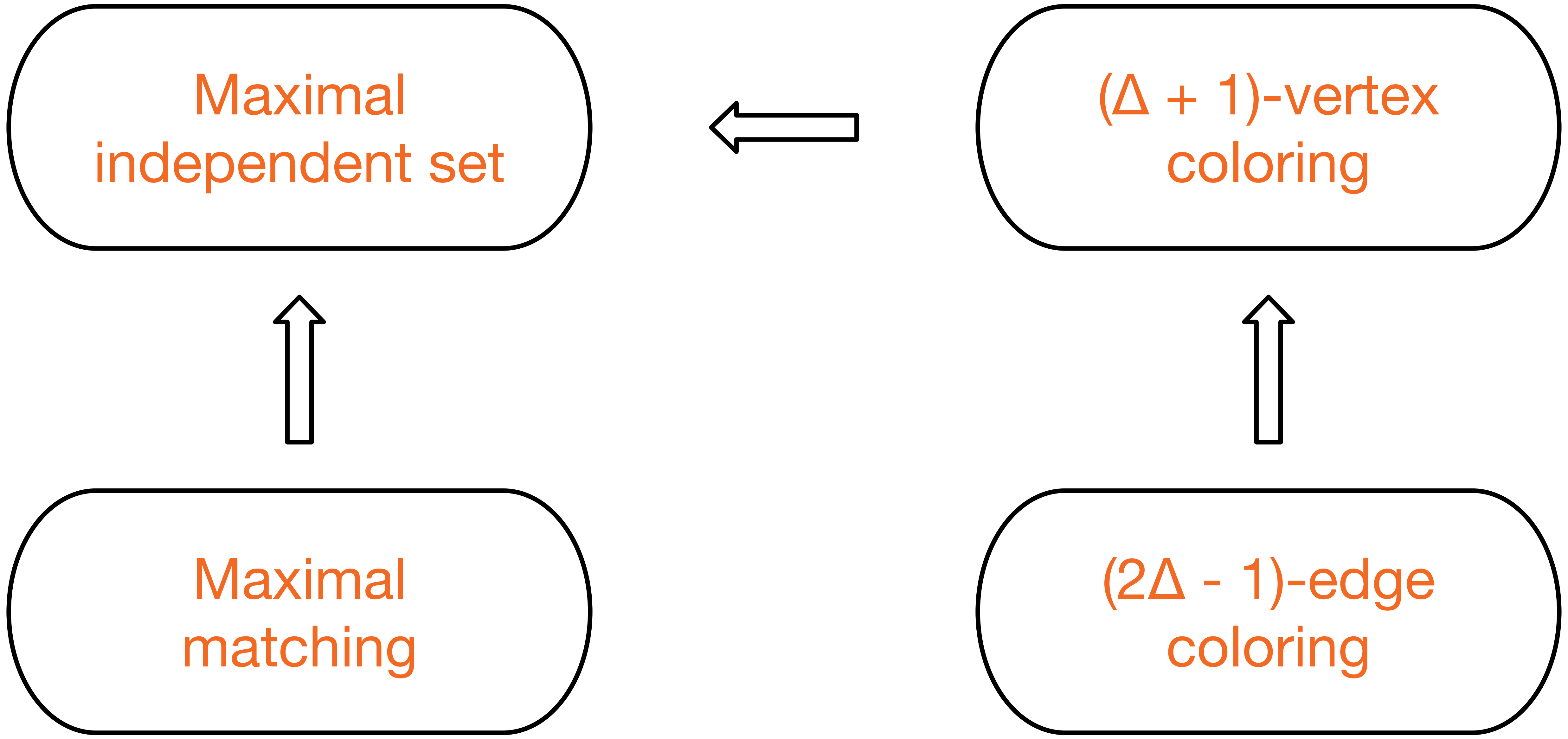


CONGEST model

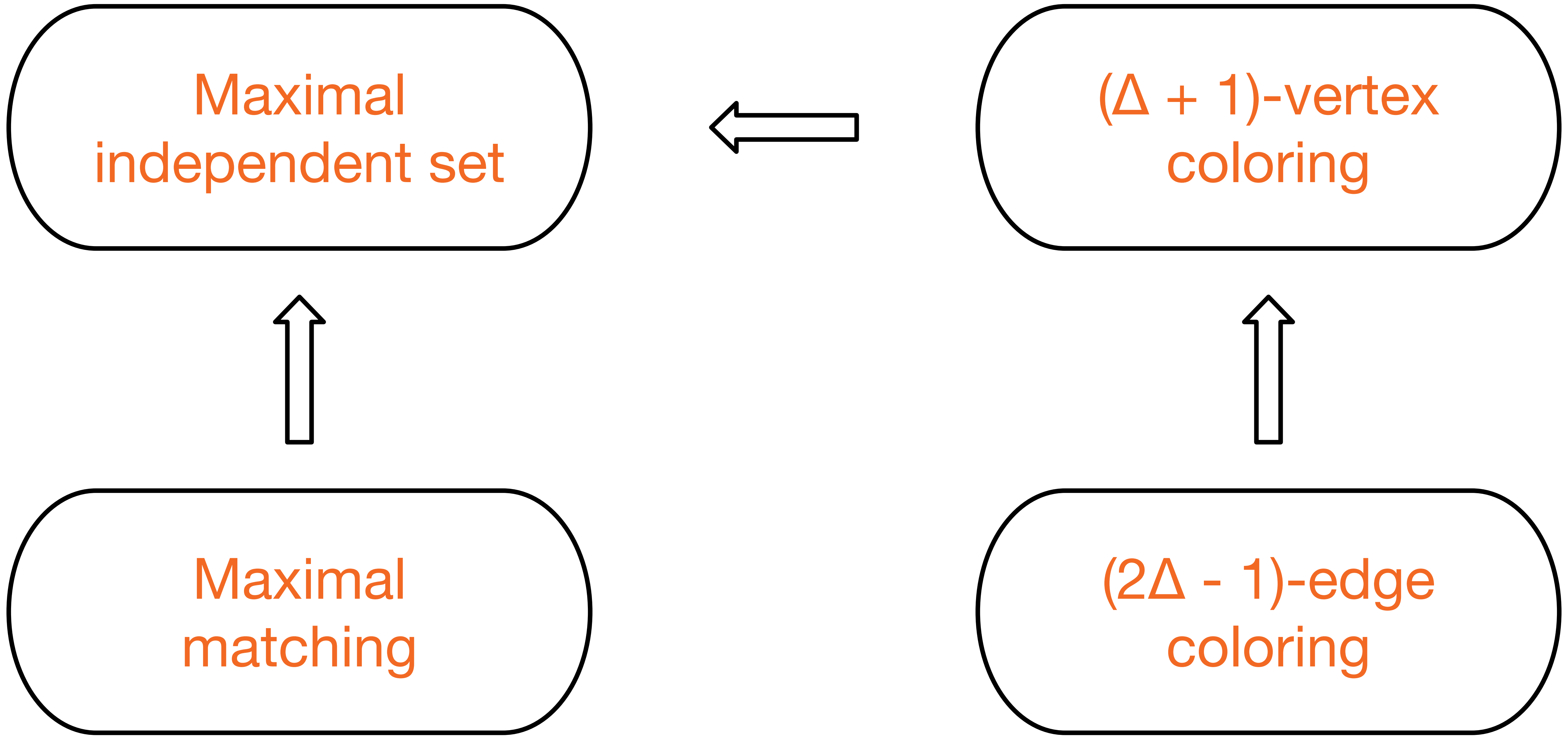
- Undirected simple graph $G = (V, E)$ of n nodes and maximum degree Δ
- Each node has a unique ID
- Synchronous message passing model
- Unbounded computation
- O(log n)-bit messages



Four classical problems

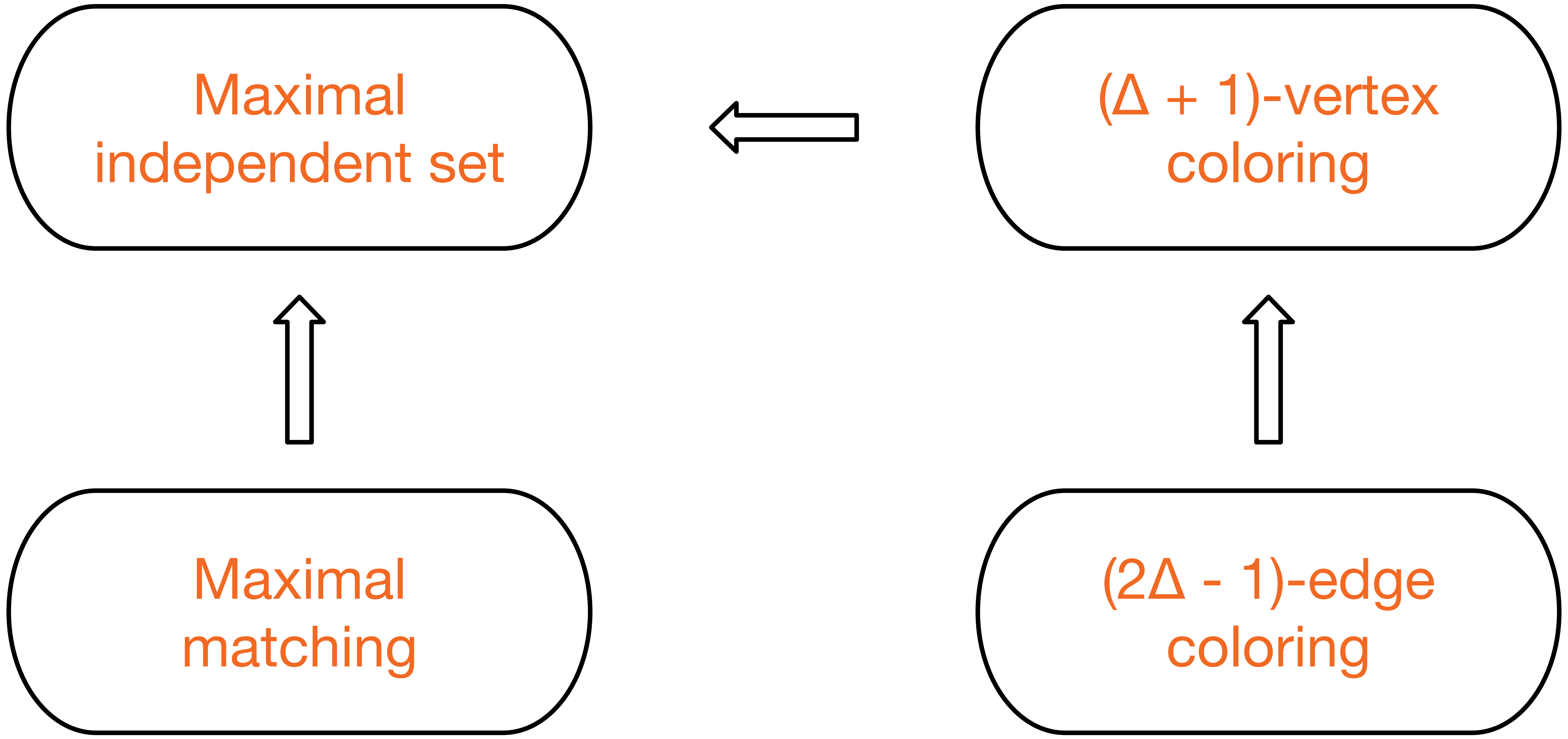


Four classical problems



These problems can be solved in $\mathrm{poly}\log n$ rounds [Rozhon, Ghaffari ’20]

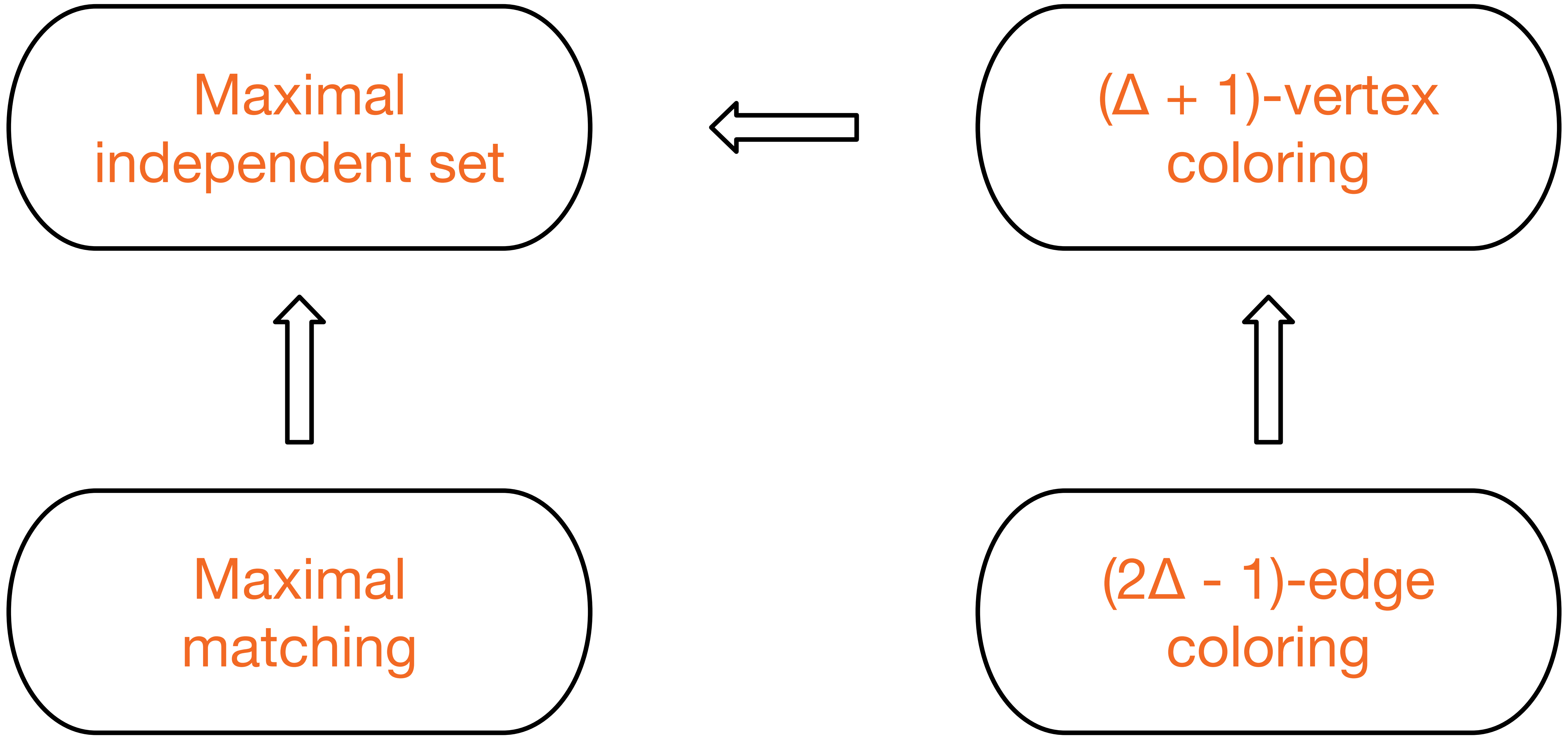
Four classical problems



These problems can be solved in $\mathrm{poly} \log n$ rounds [Rozhon, Ghaffari '20]

These problems require $\Omega(\log * n)$ rounds [Linial '87]

Four classical problems



These problems can be solved in $\mathrm{poly} \log n$ rounds [Rozhon, Ghaffari '20]

These problems require $\Omega(\log * n)$ rounds [Linial '87]

Big question: $f(\Delta) + O(\log^* n)$

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]
$(\Delta + 1)$-Vertex Coloring	$O(\sqrt{\Delta \log \Delta} + \log^* n)$ [FHK '16] [BEG '18] [MT '20]	

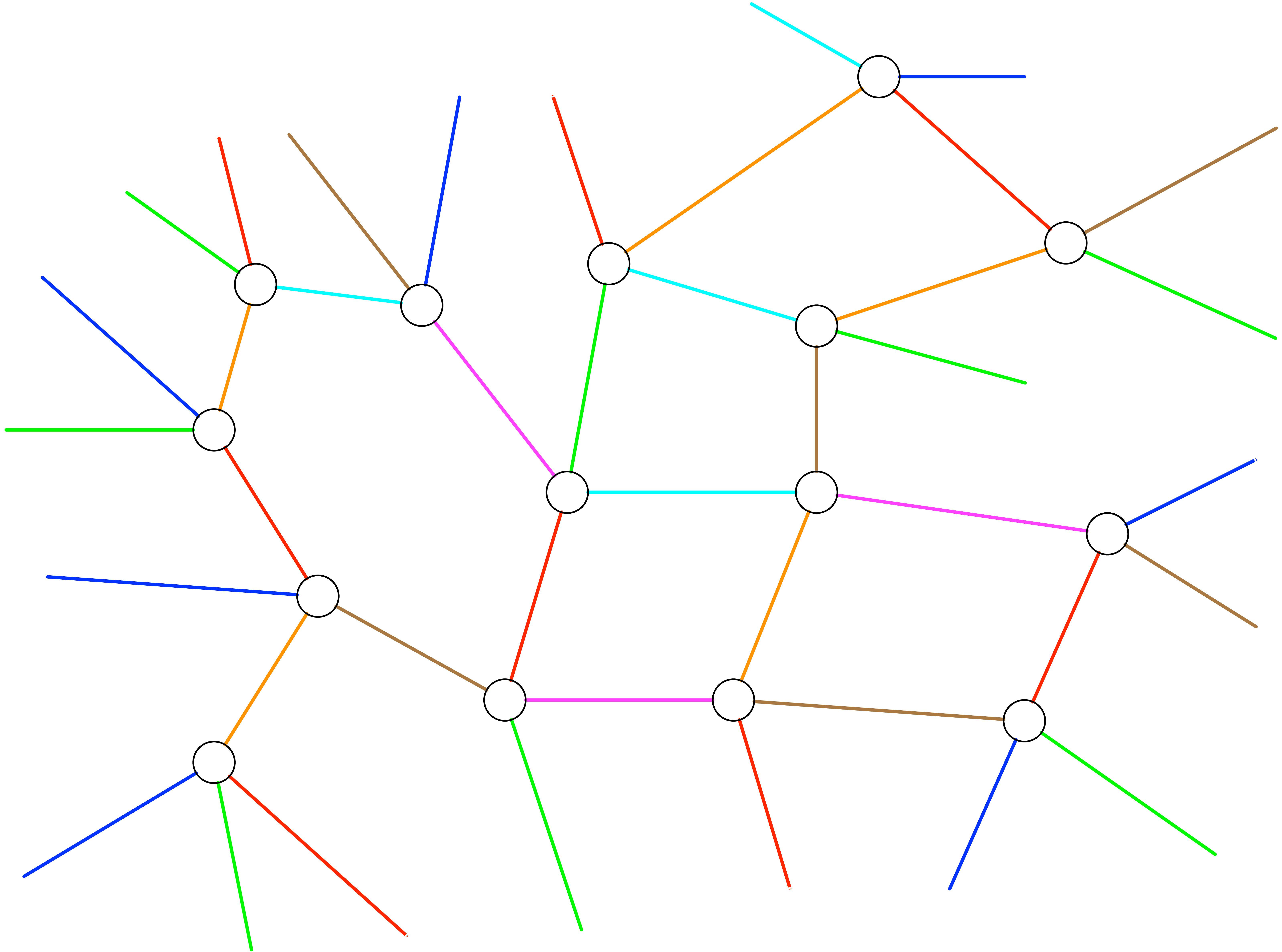
Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]
$(\Delta+1)$-Vertex Coloring	$O(\sqrt{\Delta \log \Delta} + \log^* n)$ [FHK '16] [BEG '18] [MT '20]	
$(2\Delta-1)$-Edge Coloring	$(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$ [Balliu, Kuhn, Olivetti '20]	

Four classical problems

Maximal Matching	$O(\Delta + \log^* n)$ [Panconesi, Rizzi '01]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19]
Maximal Independent Set	$O(\Delta + \log^* n)$ [Barenboim, Elkin, Kuhn '09]	$\Omega(\min\{\Delta, \log_\Delta n\})$ [BBHORS '19] [BBKO '22]
$(\Delta + 1)$-Vertex Coloring	$O(\sqrt{\Delta \log \Delta} + \log^* n)$ [FHK '16] [BEG '18] [MT '20]	
$(2\Delta - 1)$-Edge Coloring	$(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$ [Balliu, Kuhn, Olivetti '20]	

(2Δ-1)-Edge Coloring



Edge coloring: state of the art

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin '09], [Kuhn '09]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin '09], [Kuhn '09]
  - $O(\Delta^{3/4} + \log^* n)$ [Barenboim '15]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin '09], [Kuhn '09]
  - $O(\Delta^{3/4} + \log^* n)$ [Barenboim '15]
  - $O(\sqrt{\Delta \log \Delta} + \log^* n)$ [Fraigniaud, Heinrich, Kosowski '16] [Barenboim, Elkin, Goldenberg '18] [Maus, Tonoyan '20]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin ’09], [Kuhn ’09]
  - $O(\Delta^{3/4} + \log^* n)$ [Barenboim ’15]
  - $O(\sqrt{\Delta \log \Delta} + \log^* n)$ [Fraigniaud, Heinrich, Kosowski ’16] [Barenboim, Elkin, Goldenberg ’18] [Maus, Tonoyan ’20]
- $O(\Delta)$-edge coloring: $O(\Delta^{\varepsilon} + \log^* n)$ [Barenboim, Elkin ’10]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin '09], [Kuhn '09]
  - $O(\Delta^{3/4} + \log^* n)$ [Barenboim '15]
  - $O(\sqrt{\Delta \log \Delta} + \log^* n)$ [Fraigniaud, Heinrich, Kosowski '16] [Barenboim, Elkin, Goldenberg '18] [Maus, Tonoyan '20]
- $O(\Delta)$-edge coloring: $O(\Delta^{\varepsilon} + \log^* n)$ [Barenboim, Elkin '10]
- $(2\Delta - 1)$-edge coloring in $2^{O(\sqrt{\log \Delta})} + O(\log^* n)$ [Kuhn '20]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
    - $O(\Delta + \log^* n)$ [Barenboim, Elkin '09], [Kuhn '09]
    - $O(\Delta^{3/4} + \log^* n)$ [Barenboim '15]
    - $O(\sqrt{\Delta \log \Delta} + \log^* n)$ [Fraigniaud, Heinrich, Kosowski '16] [Barenboim, Elkin, Goldenberg '18] [Maus, Tonoyan '20]
- $O(\Delta)$-edge coloring: $O(\Delta^{\varepsilon} + \log^* n)$ [Barenboim, Elkin '10]
- $(2\Delta - 1)$-edge coloring in $2^{O(\sqrt{\log \Delta})} + O(\log^* n)$ [Kuhn '20]
- $(2\Delta - 1)$-edge coloring in $(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$ [Balliu, Kuhn, Olivetti '20]

Edge coloring: state of the art

- $(2\Delta - 1)$-edge coloring (achieved through $(\Delta + 1)$-vertex coloring):
  - $O(\Delta + \log^* n)$ [Barenboim, Elkin ’09], [Kuhn ’09]
  - $O(\Delta^{3/4} + \log^* n)$ [Barenboim ’15]
  - $O(\sqrt{\Delta \log \Delta} + \log^* n)$ [Fraigniaud, Heinrich, Kosowski ’16] [Barenboim, Elkin, Goldenberg ’18] [Maus, Tonoyan ’20]
- $O(\Delta)$-edge coloring: $O(\Delta^{\varepsilon} + \log^* n)$ [Barenboim, Elkin ’10]
- $(2\Delta - 1)$-edge coloring in $2^{O(\sqrt{\log \Delta})} + O(\log^* n)$ [Kuhn ’20]
- $(2\Delta - 1)$-edge coloring in $(\log \Delta)^{O(\log \log \Delta)} + O(\log^* n)$ [Balliu, Kuhn, Olivetti ’20]
- **Can we solve $(2\Delta - 1)$-edge coloring in $\operatorname{poly} \log \Delta + O(\log^* n)$ rounds?**

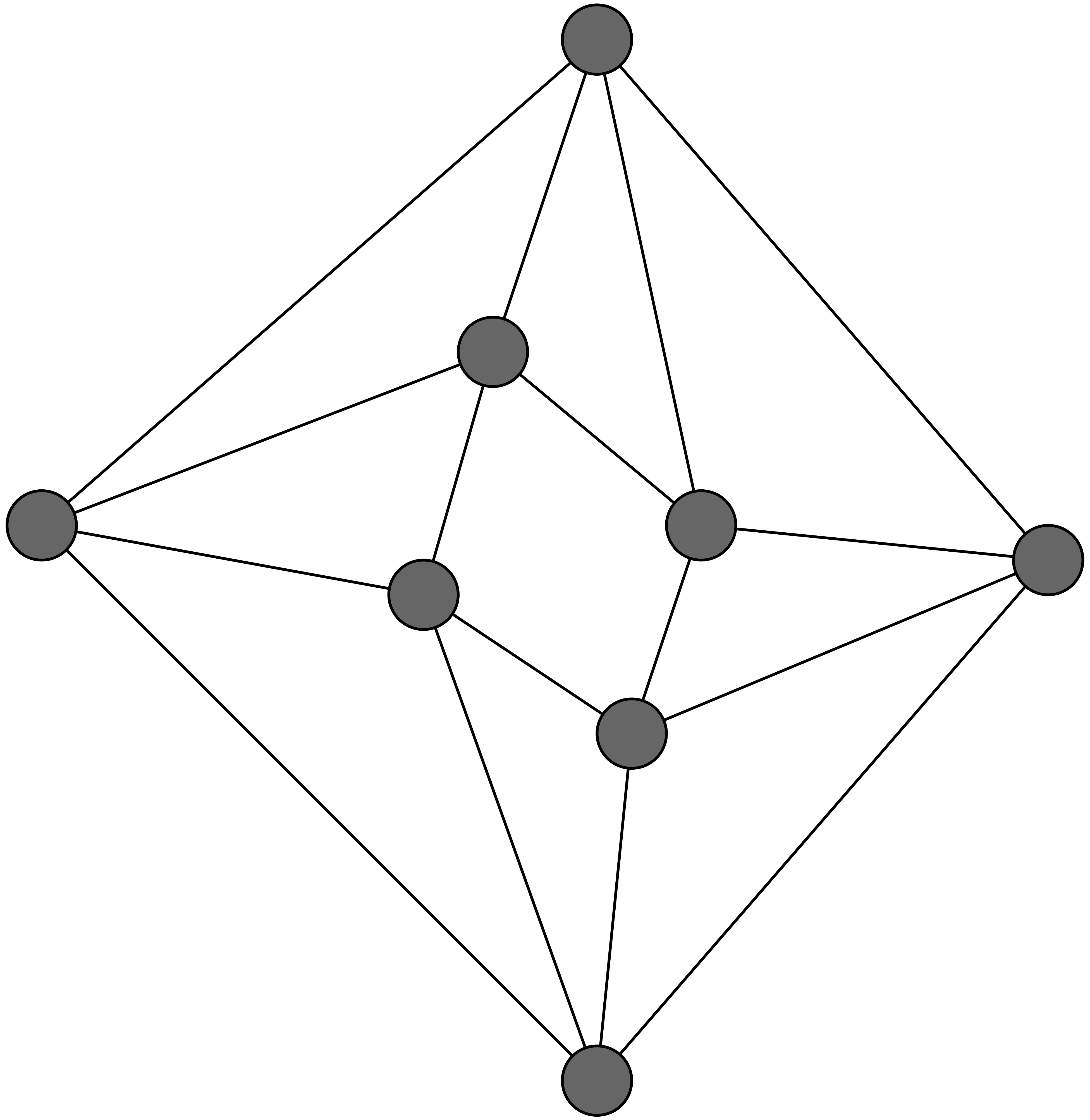
Our results

$(2\Delta - 1)$-Edge Coloring	$O(\text{poly}\log\Delta + \log^* n)$	LOCAL model
$O(\Delta)$-Edge Coloring	$O(\text{poly}\log\Delta + \log^* n)$	CONGEST model

Our results

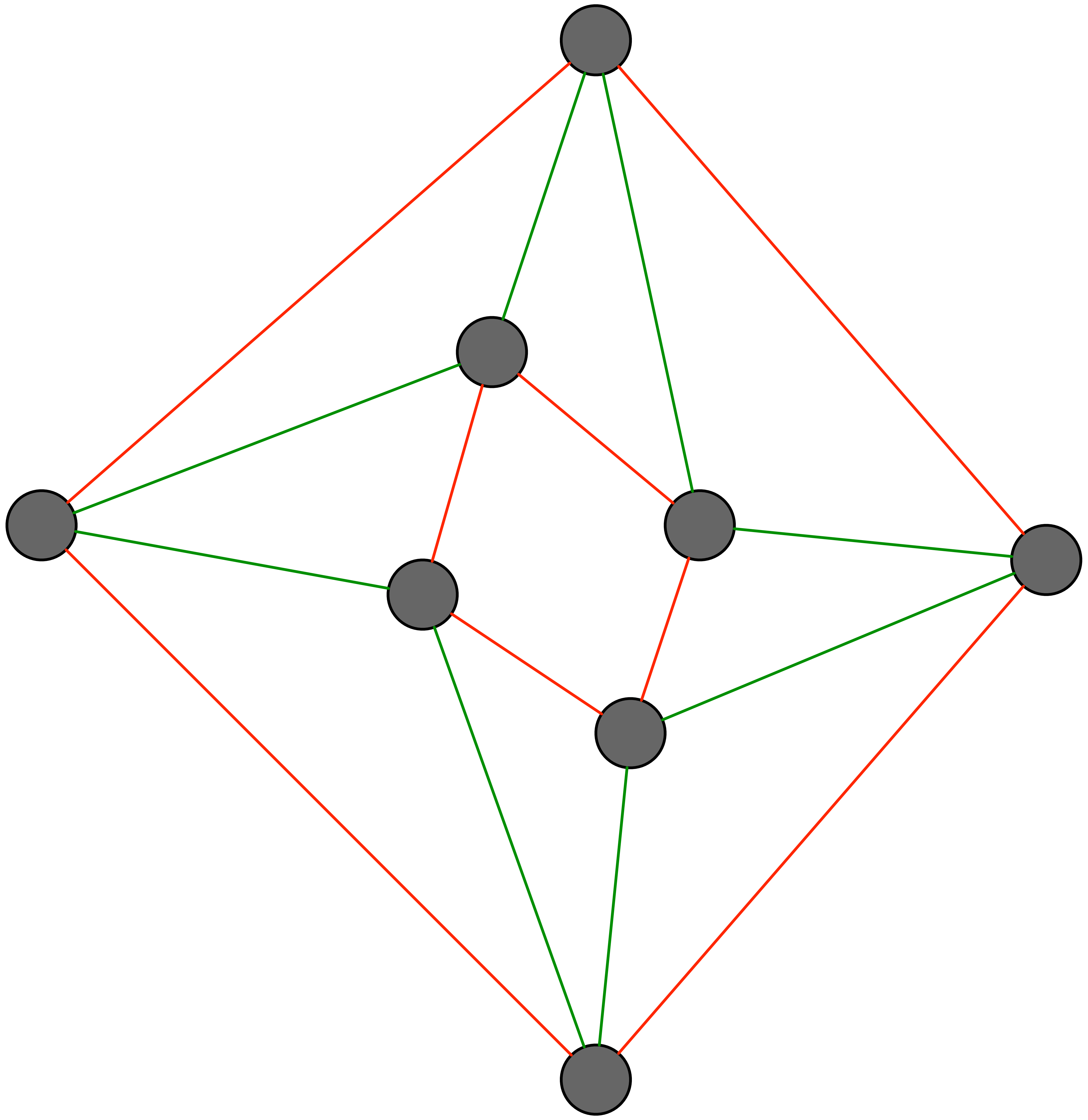
$(\text{degree}+1)$-List Edge Coloring	$O(\log^7 C \cdot \log^5 \Delta + \log^* n)$	LOCAL model
$(2\Delta-1)$-Edge Coloring	$O(\log^{12} \Delta + \log^* n)$	LOCAL model
$(8+\varepsilon)\Delta$-Edge Coloring	$O\left(\frac{\log^{12} \Delta}{\varepsilon^6} + \log^* n\right)$	CONGEST model

A possible approach

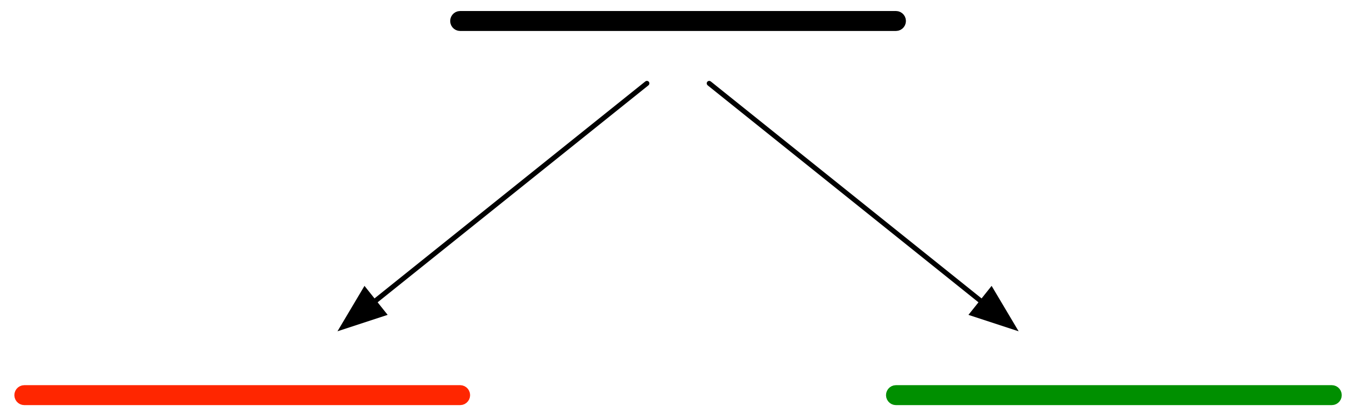


Try to **recursively color** the edges with **2 colors**, such that each node has roughly the **same amount of incident edges for each color**

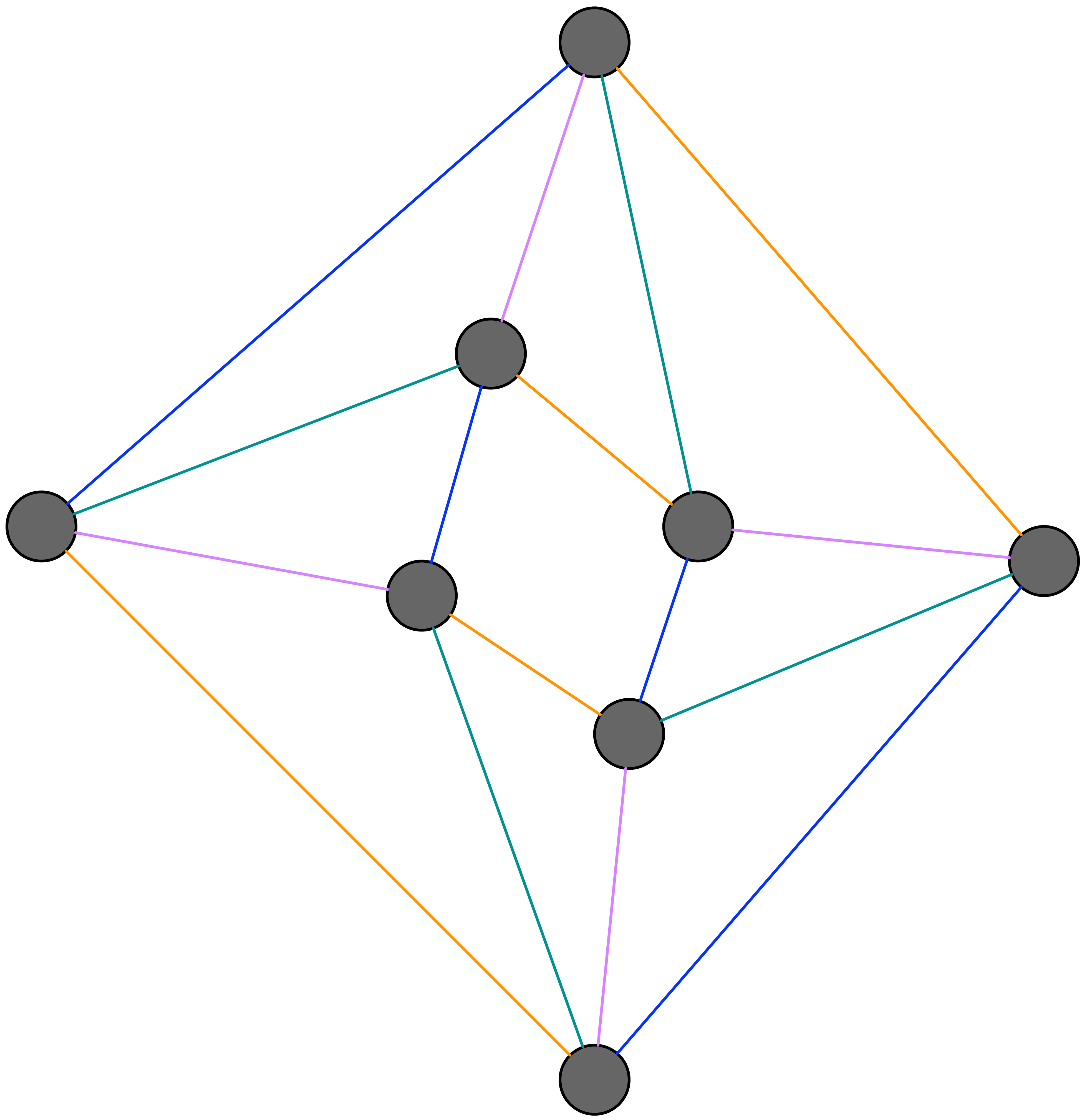
A possible approach



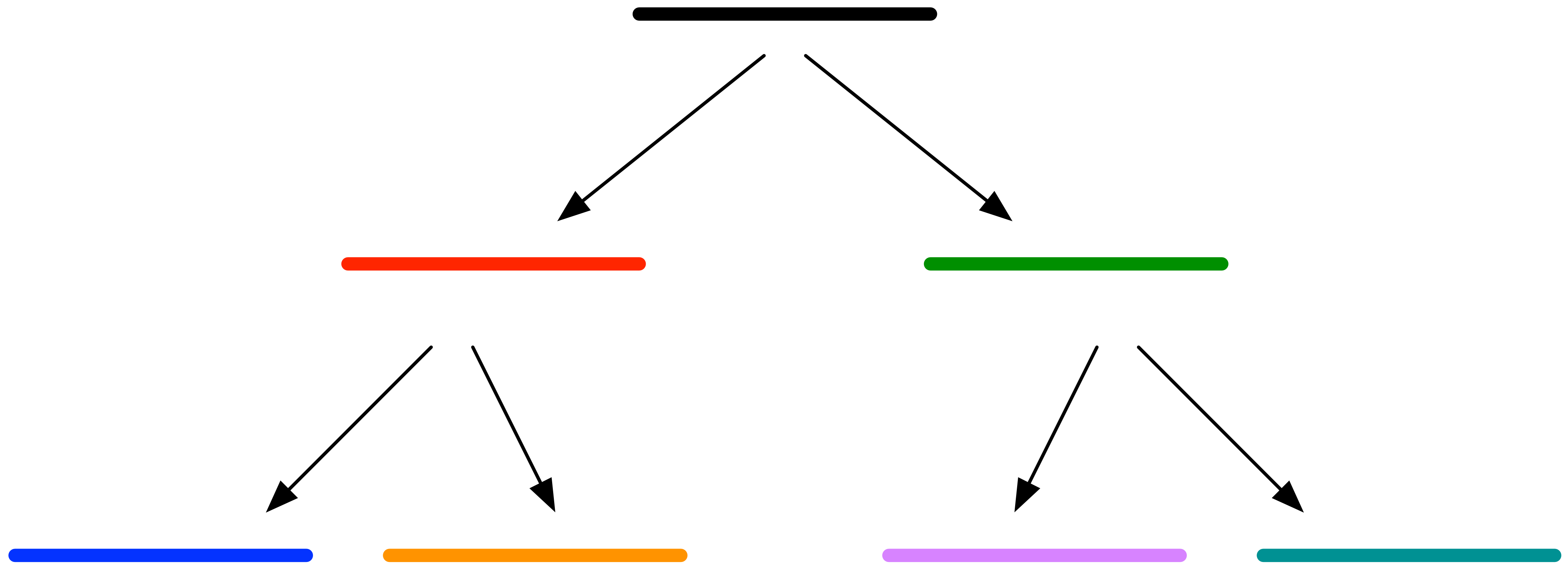
Try to **recursively color** the edges with **2 colors**, such that each node has roughly the **same amount of incident edges for each color**



A possible approach



Try to **recursively color** the edges with **2 colors**, such that each node has roughly the **same amount of incident edges for each color**



A possible approach

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.

- After T steps of recursion the maximum degree is $\Delta \cdot \left(\frac{1+\varepsilon}{2}\right)^{T}$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.

- After $\mathbf{T}$ steps of recursion the maximum degree is $\mathbf{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$

- Let us fix $\mathbf{T} = \log \mathbf{\Delta}$ and $\varepsilon = \mathbf{1}/\log \mathbf{\Delta}$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\Delta \cdot \left(\frac{1+\varepsilon}{2}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \Delta$ and $\varepsilon = 1/\log \Delta$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\mathbf{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \mathbf{\Delta}$ and $\varepsilon = \mathbf{1}/\log \mathbf{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \boldsymbol{\Delta}$ and $\varepsilon = \mathbf{1}/\log \boldsymbol{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^{T} \cdot O(1) = O(\Delta)$
- **Running time:** $\log \Delta \cdot T_{balanced\_2\_col} + T_{\text{final}}$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.

- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$

- Let us fix $\mathbf{T} = \log \boldsymbol{\Delta}$ and $\varepsilon = \mathbf{1}/\log \boldsymbol{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$
- **Running time:** $\log \Delta \cdot T_{balanced\_2\_col} + T_{\text{final}}$

Can be done in just $O(\log^* n)$

A possible approach

- Start from a graph of **maximum degree Δ, 2-color** the edges such that the graph induced by each color has **maximum degree roughly Δ/2, Recurse** on each subgraph.
- After $\mathbf{T}$ steps of recursion the maximum degree is $\boldsymbol{\Delta} \cdot \left(\frac{\mathbf{1}+\varepsilon}{\mathbf{2}}\right)^{\mathbf{T}}$
- Let us fix $\mathbf{T} = \log \boldsymbol{\Delta}$ and $\varepsilon = \mathbf{1}/\log \boldsymbol{\Delta}$
- After $\mathbf{T}$ steps each subgraph has **constant maximum degree**
- **The number of colors is** $2^T \cdot O(1) = O(\Delta)$
- **Running time:** $\log \Delta \cdot T_{balanced\_2\_col} + T_{\text{final}}$

This requires too much!

Can be done in just $O(\log^* n)$

A possible approach: the issue

- **2-coloring** the edges such that each node has **at least one incident edge for each color** requires $\Omega(\log n)$ rounds, but our target runtime is $O(\text{poly} \log \Delta + \log^* n)$

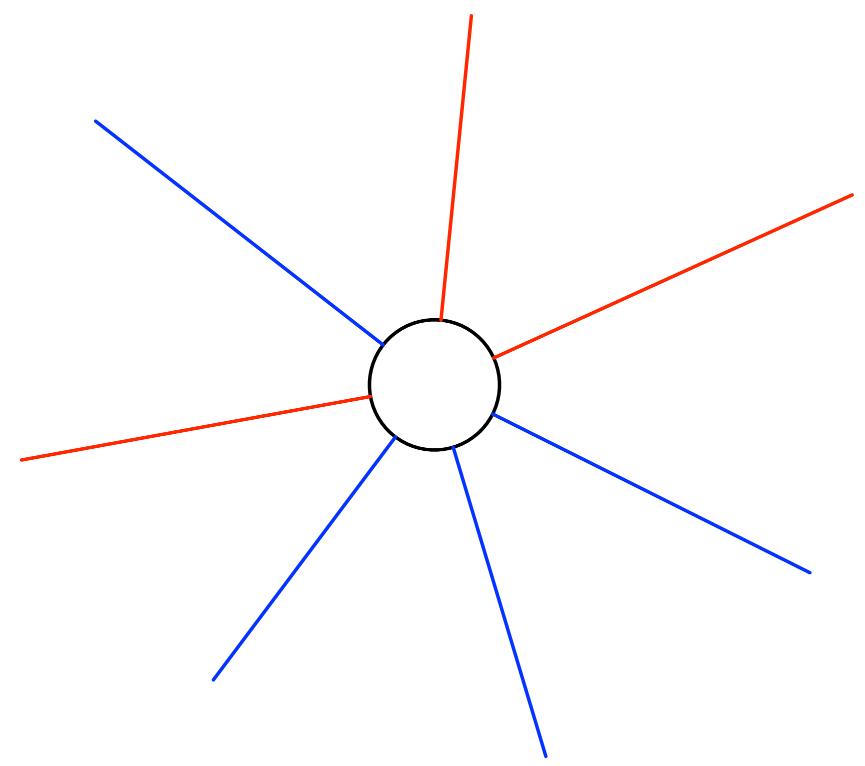
Relaxing the problem: different splitting

Relaxing the problem: different splitting

- d-**node**-defective c-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.

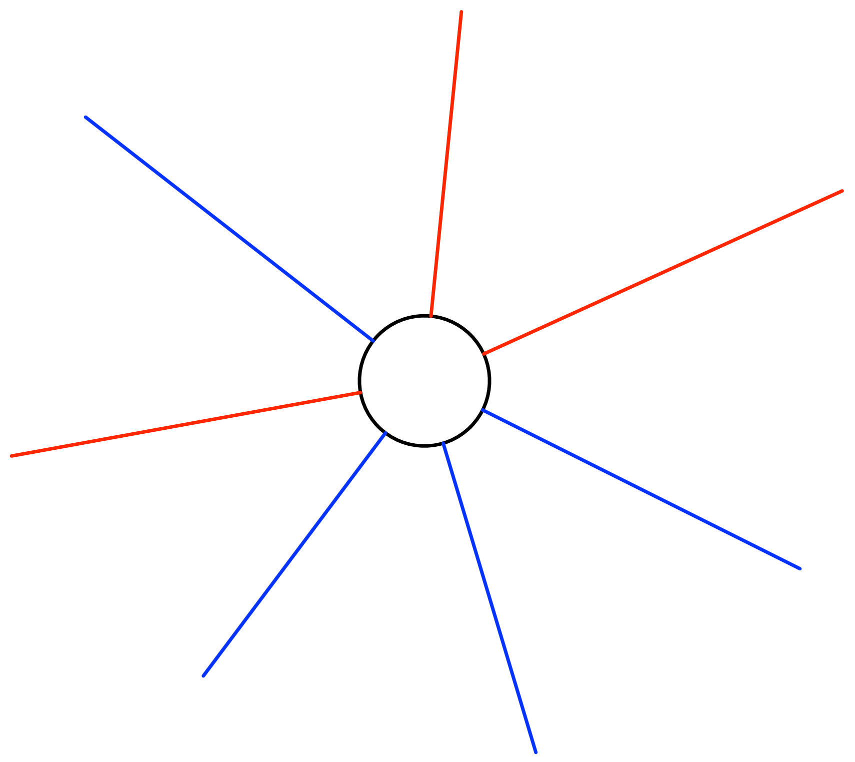
Relaxing the problem: different splitting

- d-**node**-defective c-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.



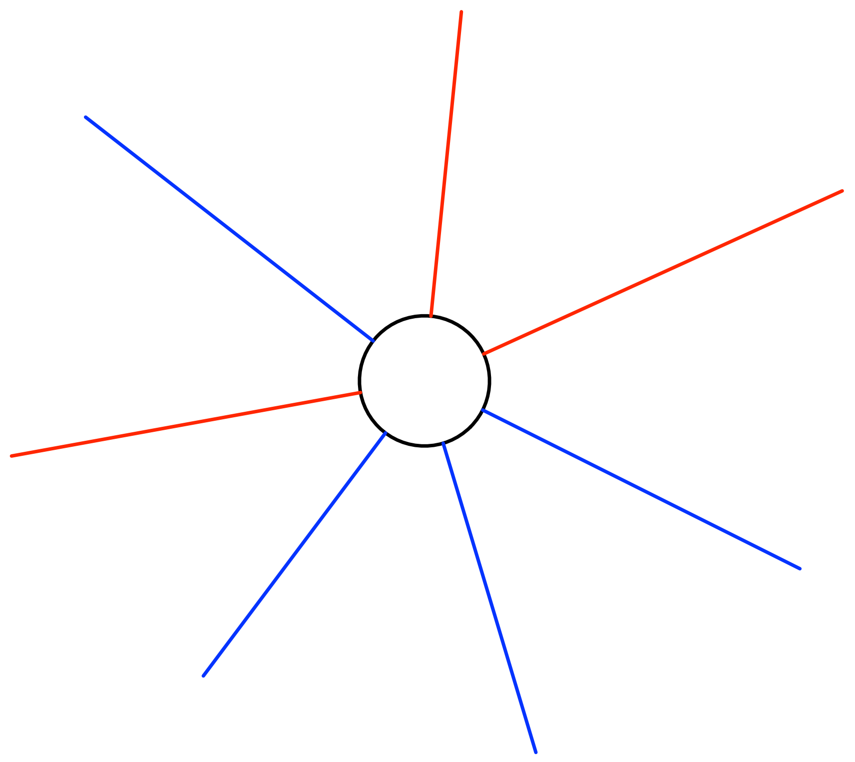
Relaxing the problem: different splitting

- **d-node**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.



This is **hard** for $c = 2$ and $d \leq \Delta - 1$

Relaxing the problem: different splitting



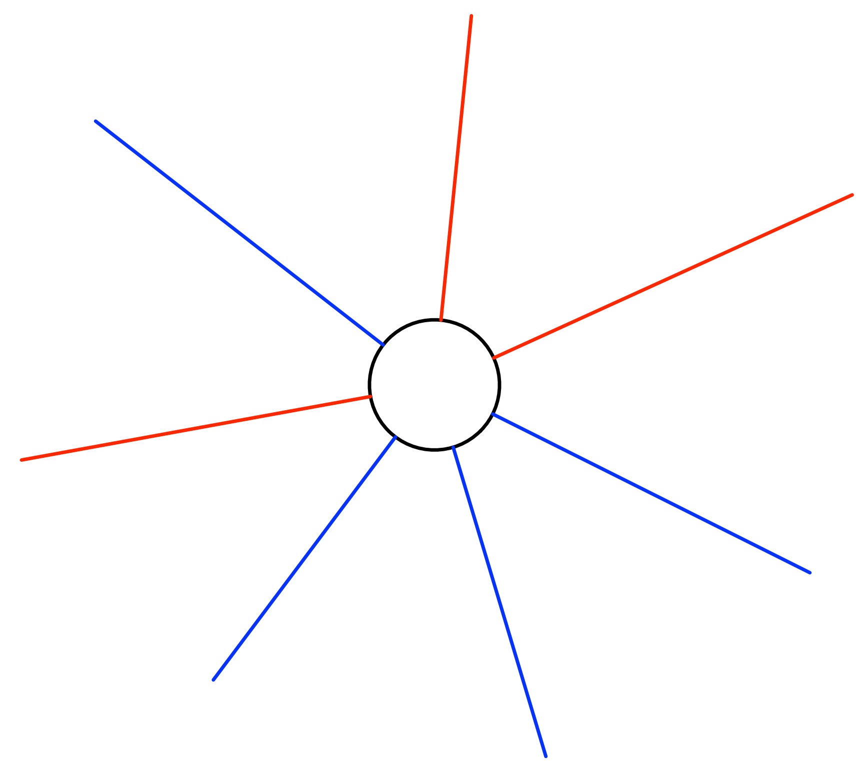
- **d-node**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.

  This is **hard** for $c = 2$ and $d \leq \Delta - 1$

  Also, this seems to be useful only for $d \leq (1 + \varepsilon)\Delta / c$

Relaxing the problem: different splitting

- **d-node**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.

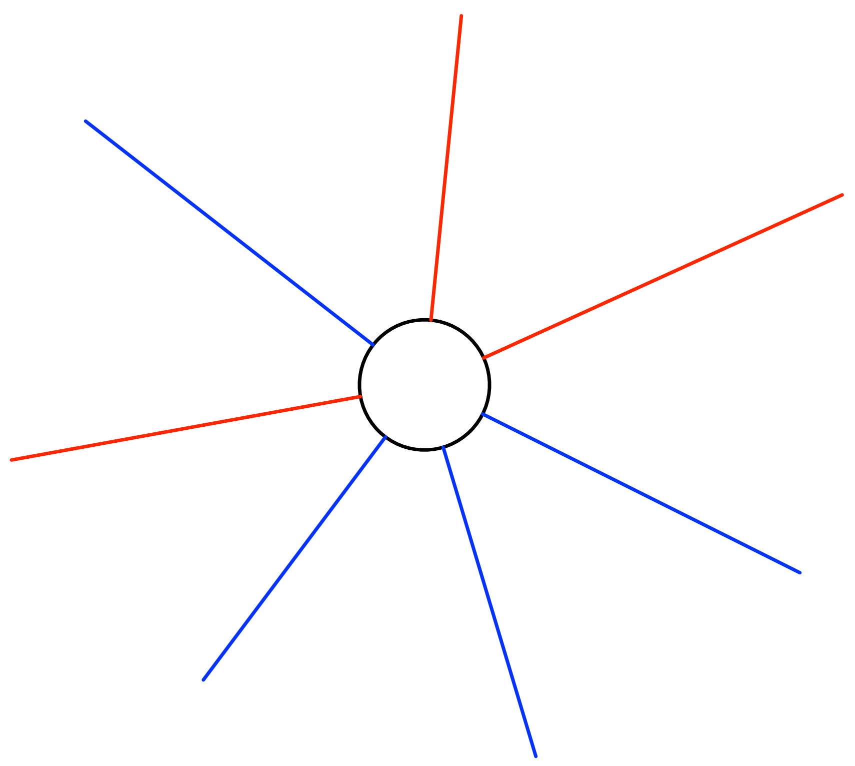


This is **hard** for $c = 2$ and $d \leq \Delta - 1$

Also, this seems to be useful only for $d \leq (1 + \varepsilon)\Delta / c$

Actually, this is hard for all parameters for which it is useful!

Relaxing the problem: different splitting



- d-**node**-defective c-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.

  This is **hard** for $c = 2$ and $d \leq \Delta - 1$

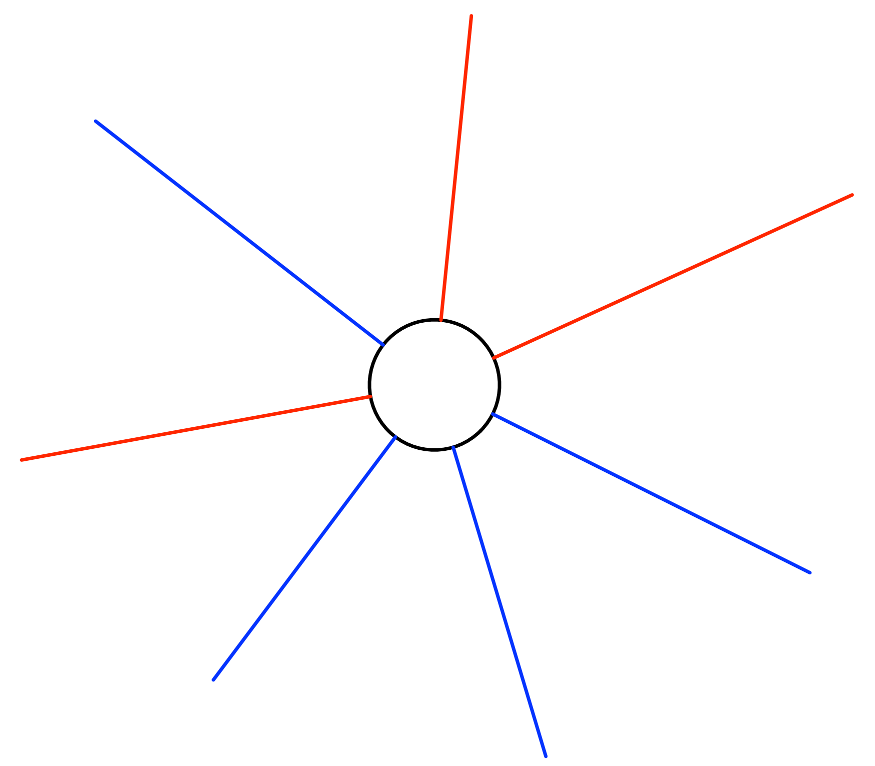
  Also, this seems to be useful only for $d \leq (1 + \varepsilon)\Delta / c$

  Actually, this is hard for all parameters for which it is useful!

- d-**edge**-defective c-edge-coloring: **color the edges** with **c** colors such that **each edge** has at most **d** incident **edges of the same color**.

Relaxing the problem: different splitting

- **d**-**node**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.

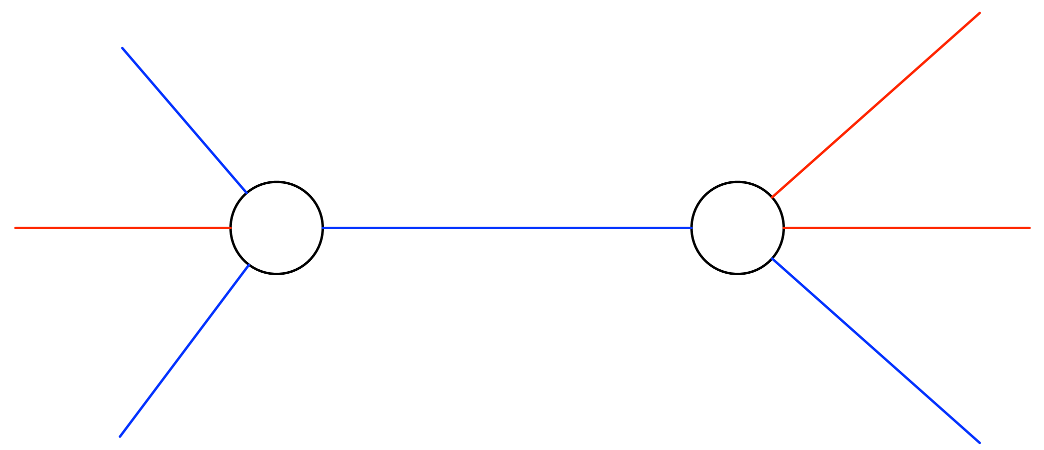


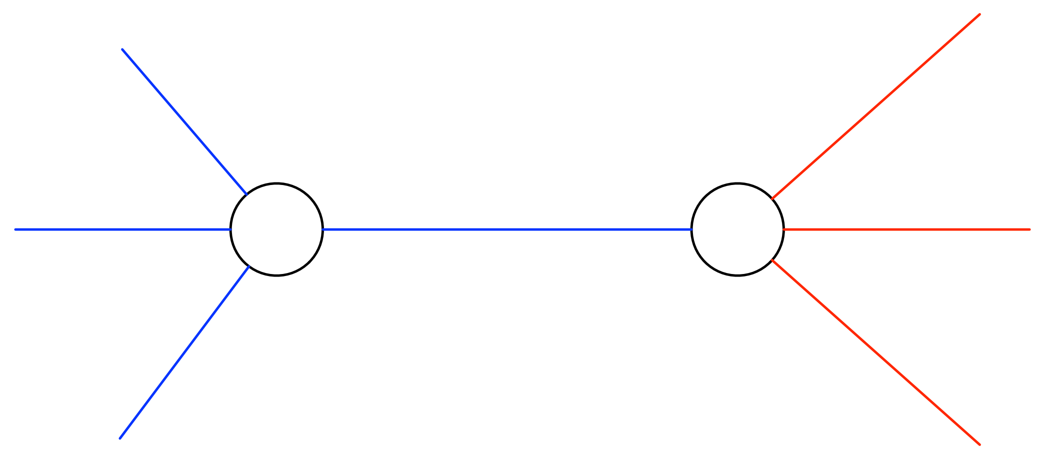
  This is **hard** for $c = 2$ and $d \leq \Delta - 1$

  Also, this seems to be useful only for $d \leq (1 + \varepsilon)\Delta / c$

  Actually, this is hard for all parameters for which it is useful!

- **d**-**edge**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each edge** has at most **d** incident **edges of the same color**.





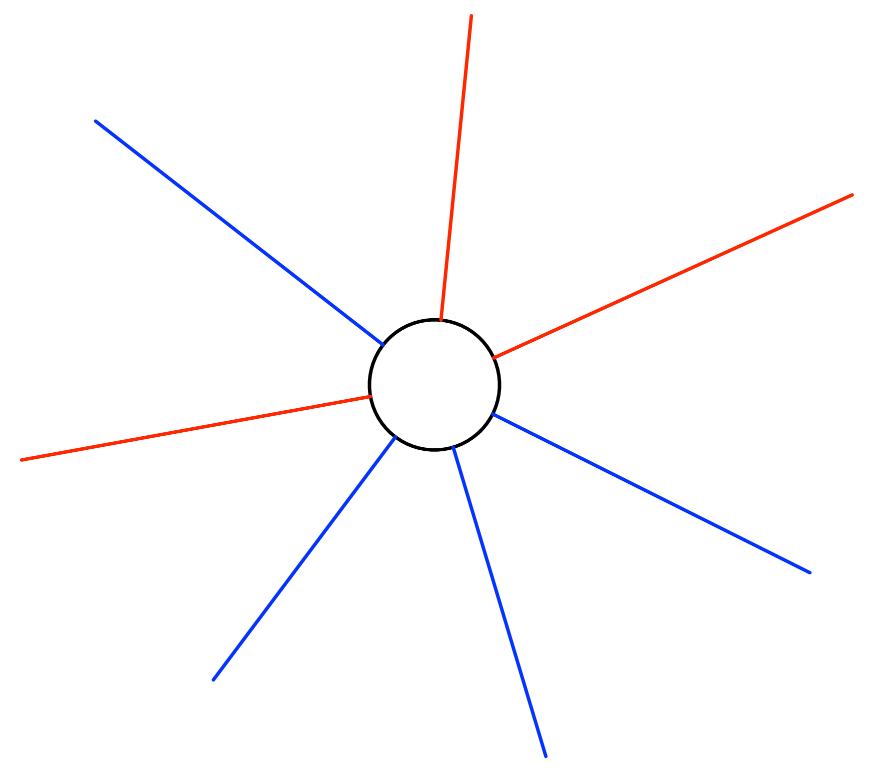
Relaxing the problem: different splitting

- **d-node**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each node** has at most **d** incident **edges of the same color**.

  This is **hard** for $c = 2$ and $d \leq \Delta - 1$

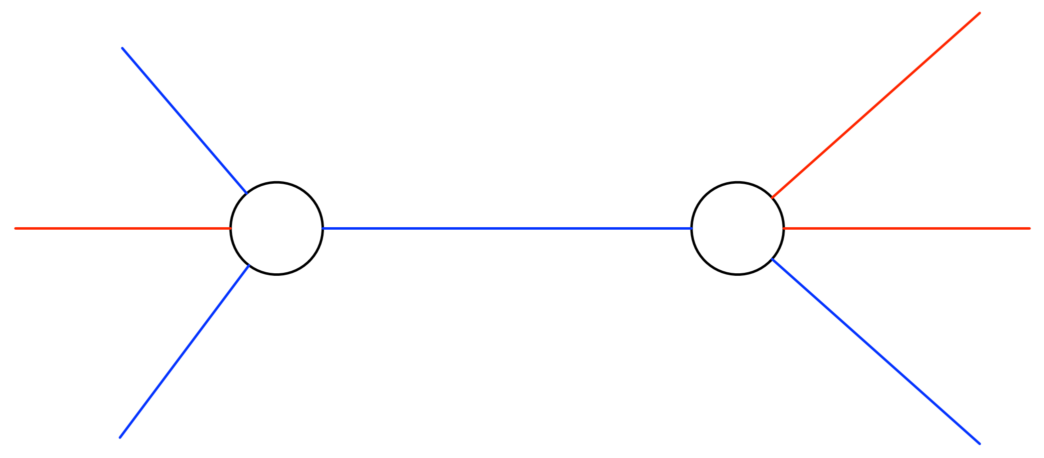
  Also, this seems to be useful only for $d \leq (1 + \varepsilon)\Delta / c$

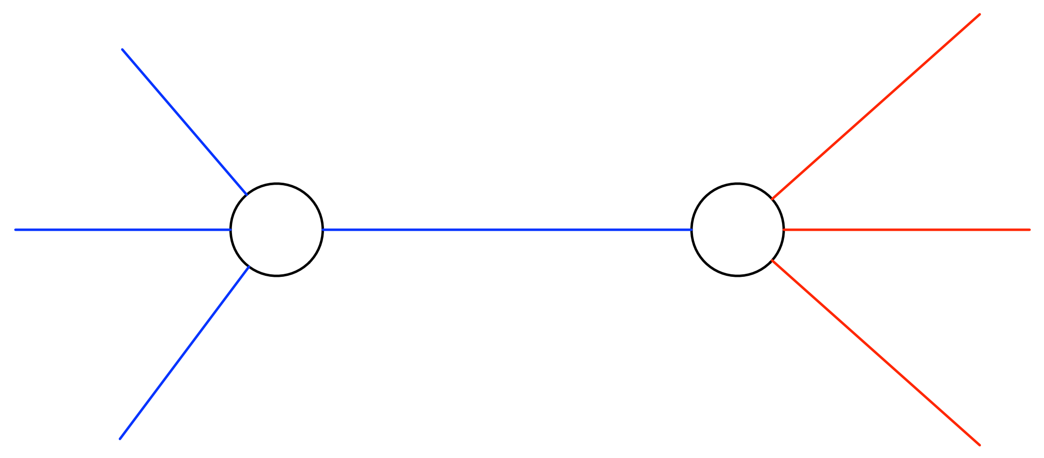
  Actually, this is hard for all parameters for which it is useful!



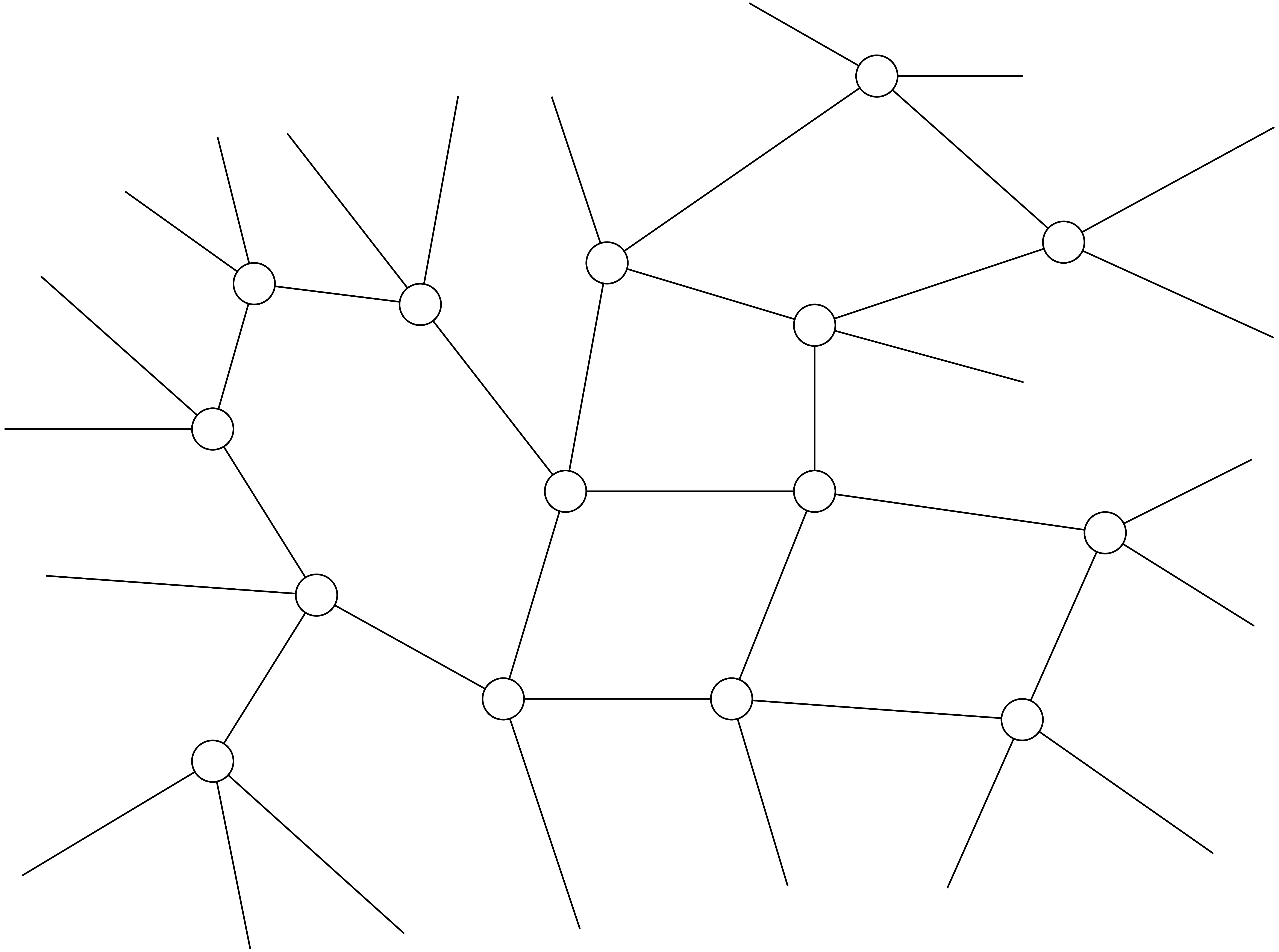
- **d-edge**-defective **c**-edge-coloring: **color the edges** with **c** colors such that **each edge** has at most **d** incident **edges of the same color**.

  This is the problem that we tried to solve, for $c = 2$ and $d \leq (1 + \varepsilon)\Delta$

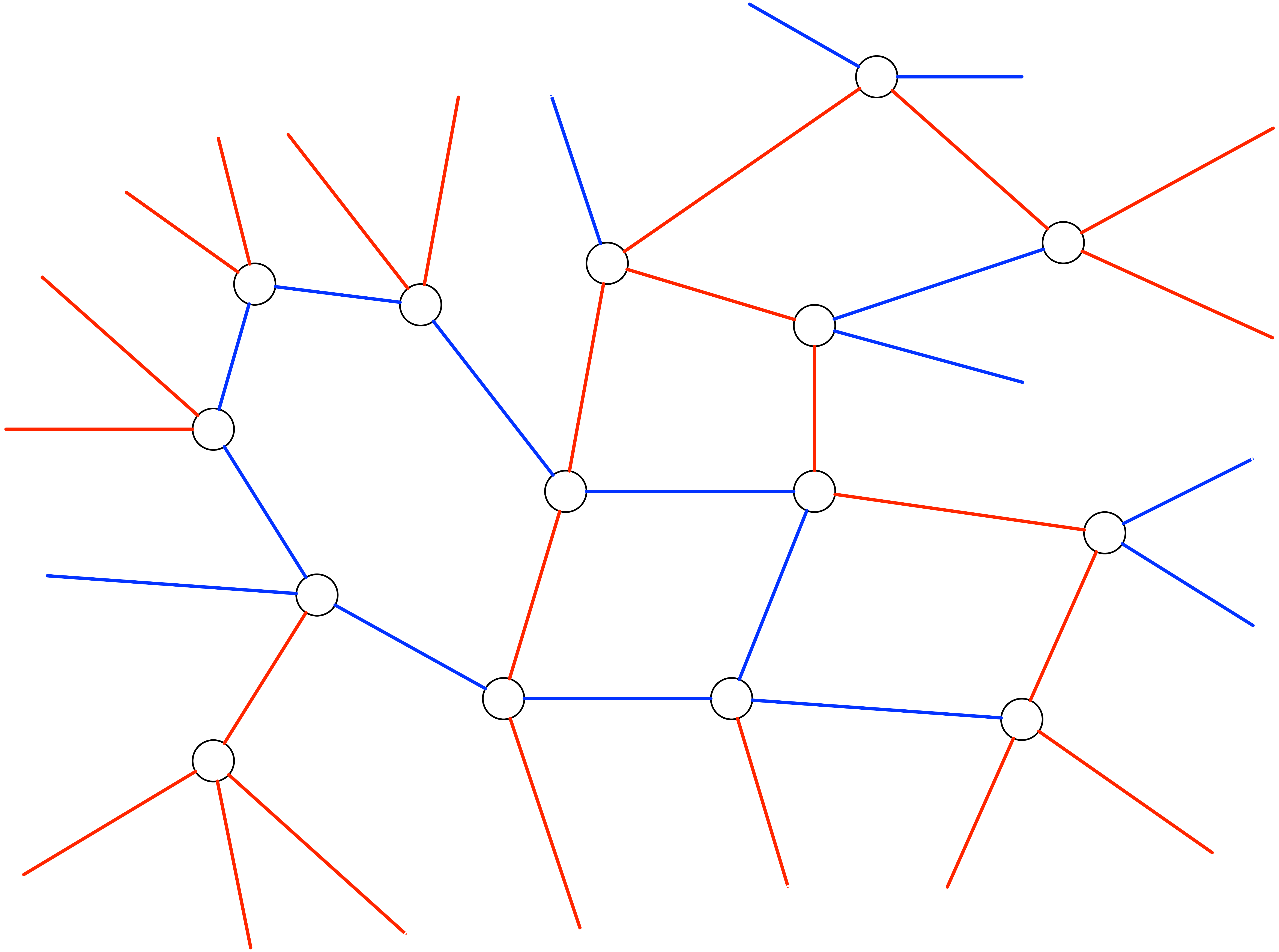




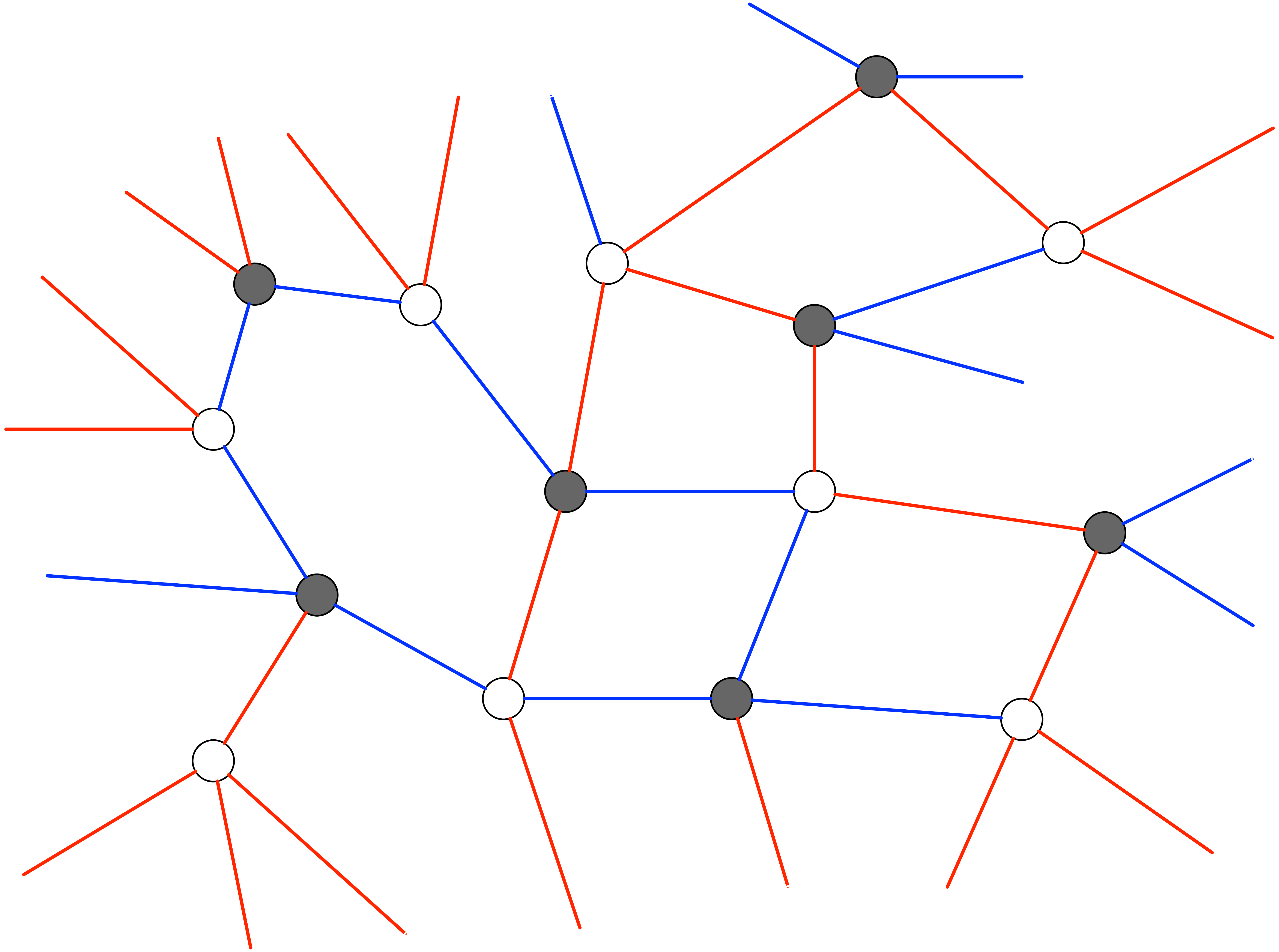
Edge Defective Edge Coloring



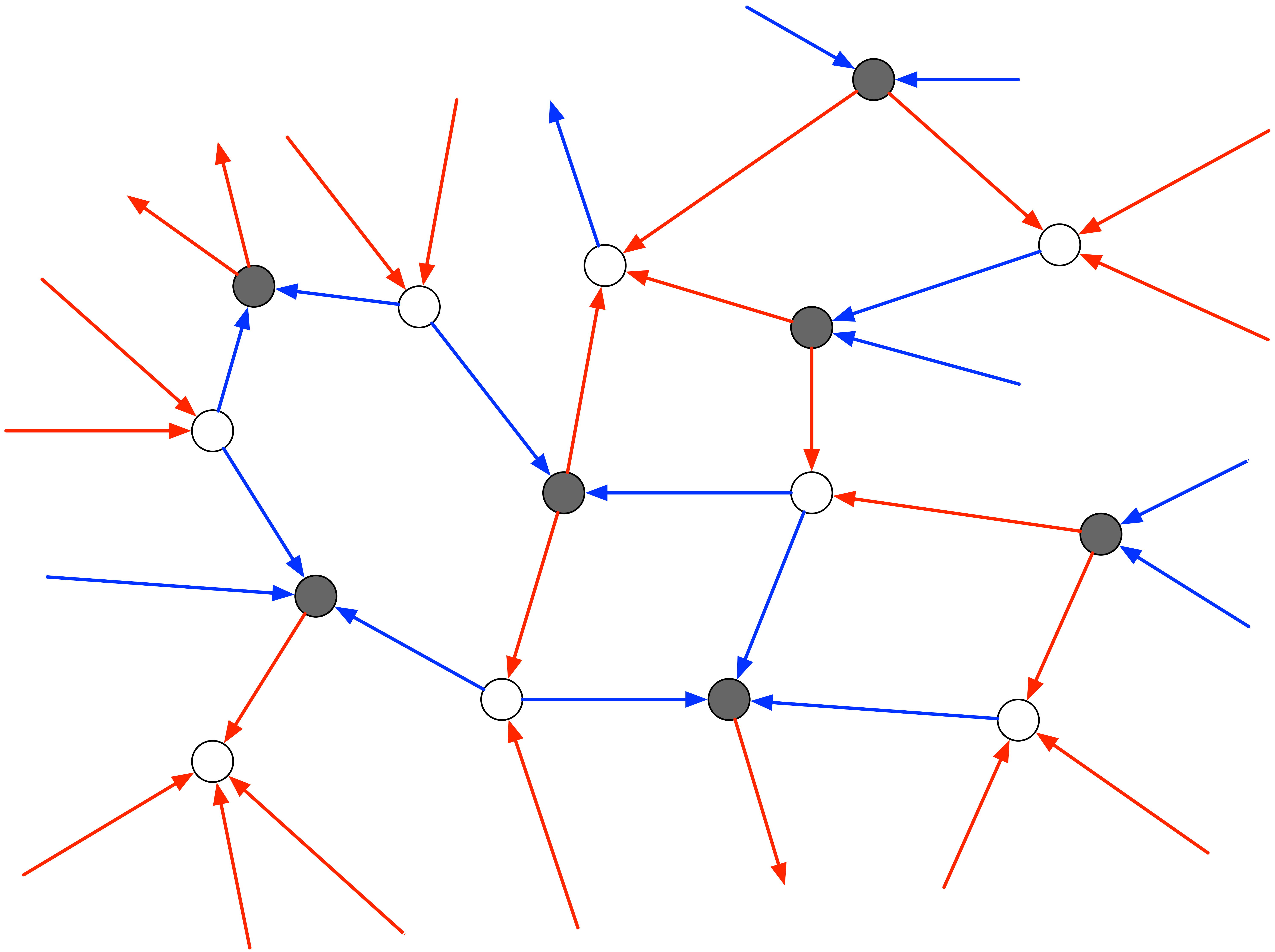
Edge Defective Edge Coloring



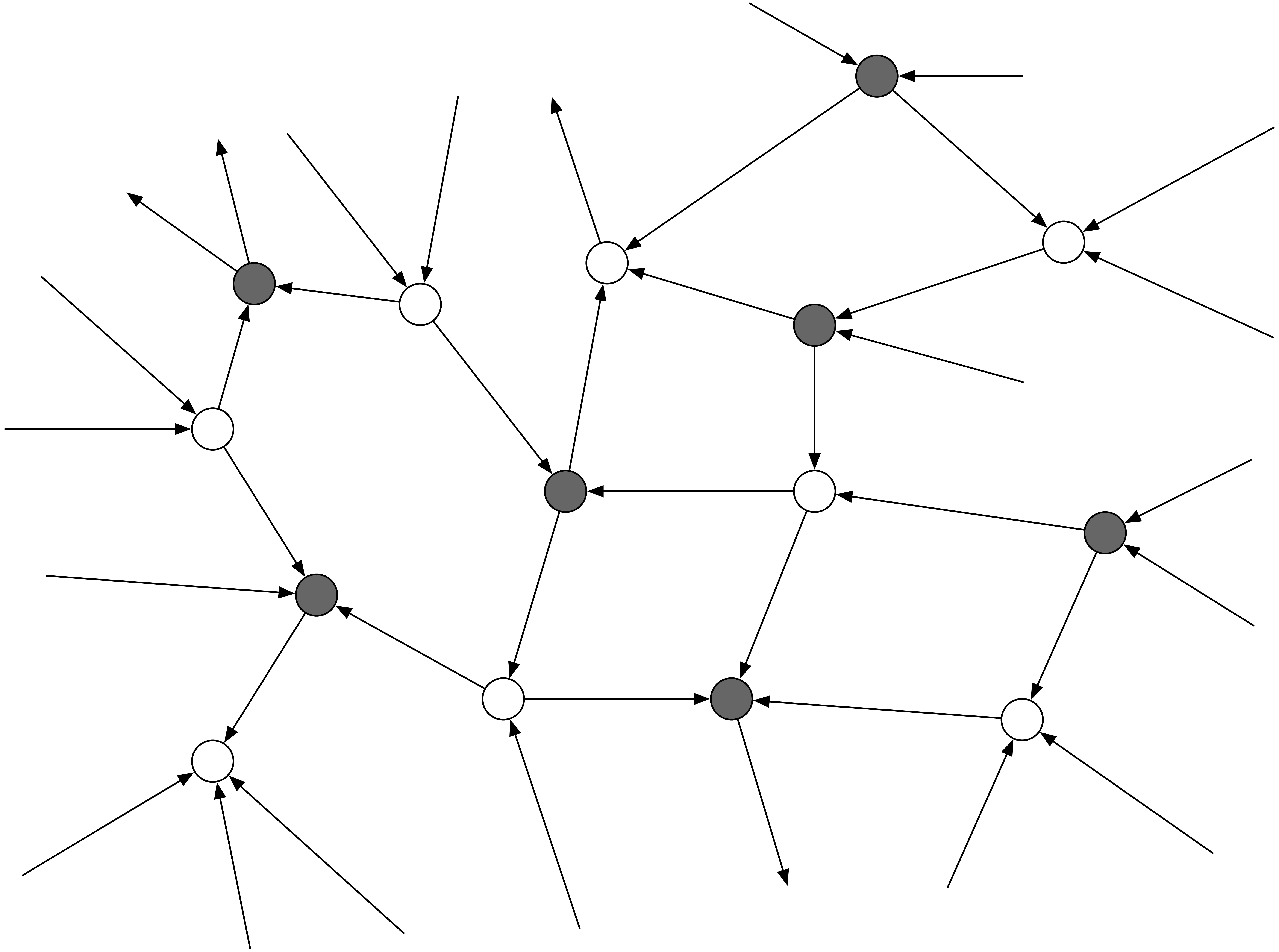
Edge Defective Edge Coloring



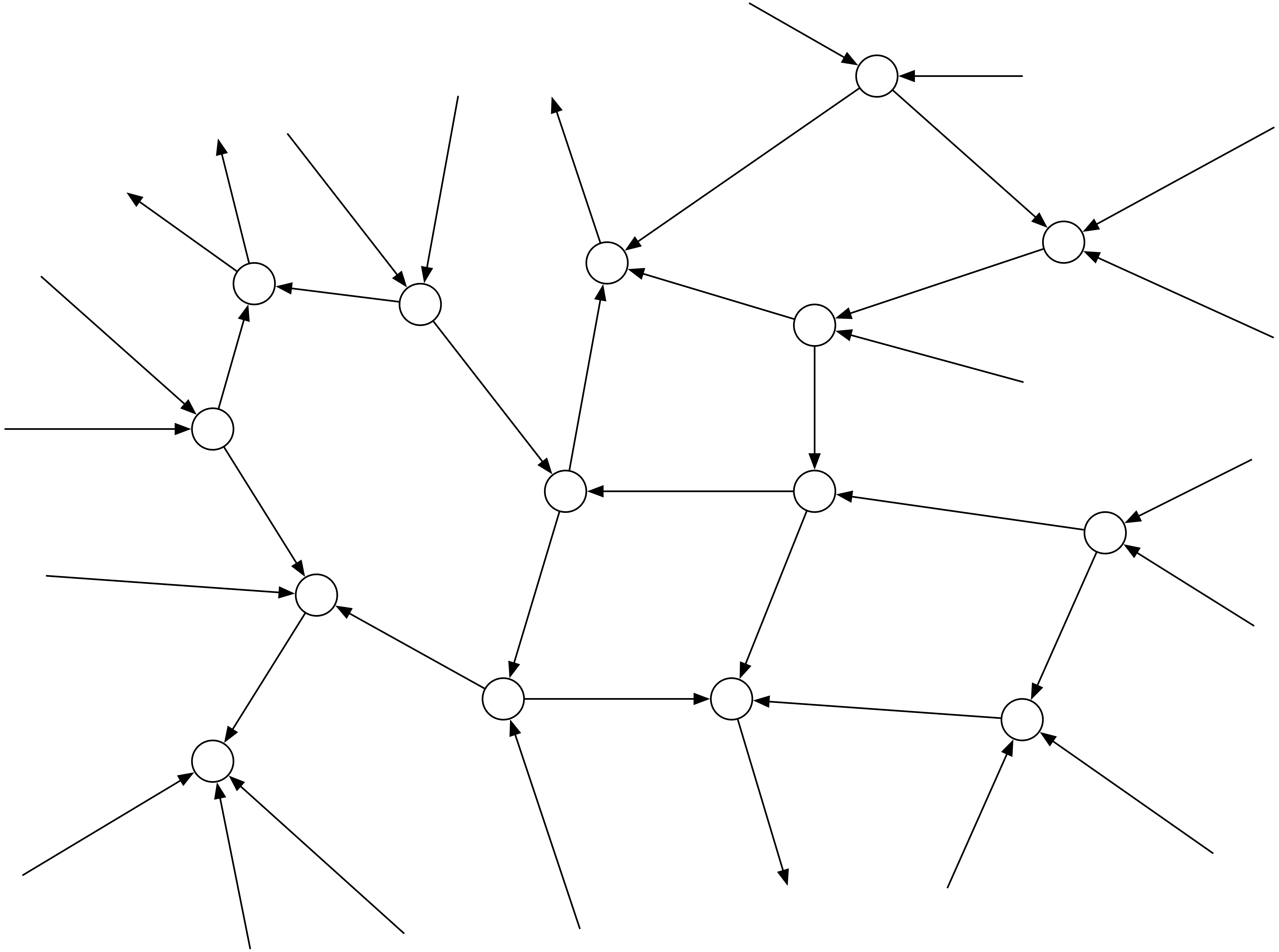
Edge Defective Edge Coloring



Edge Defective Edge Coloring

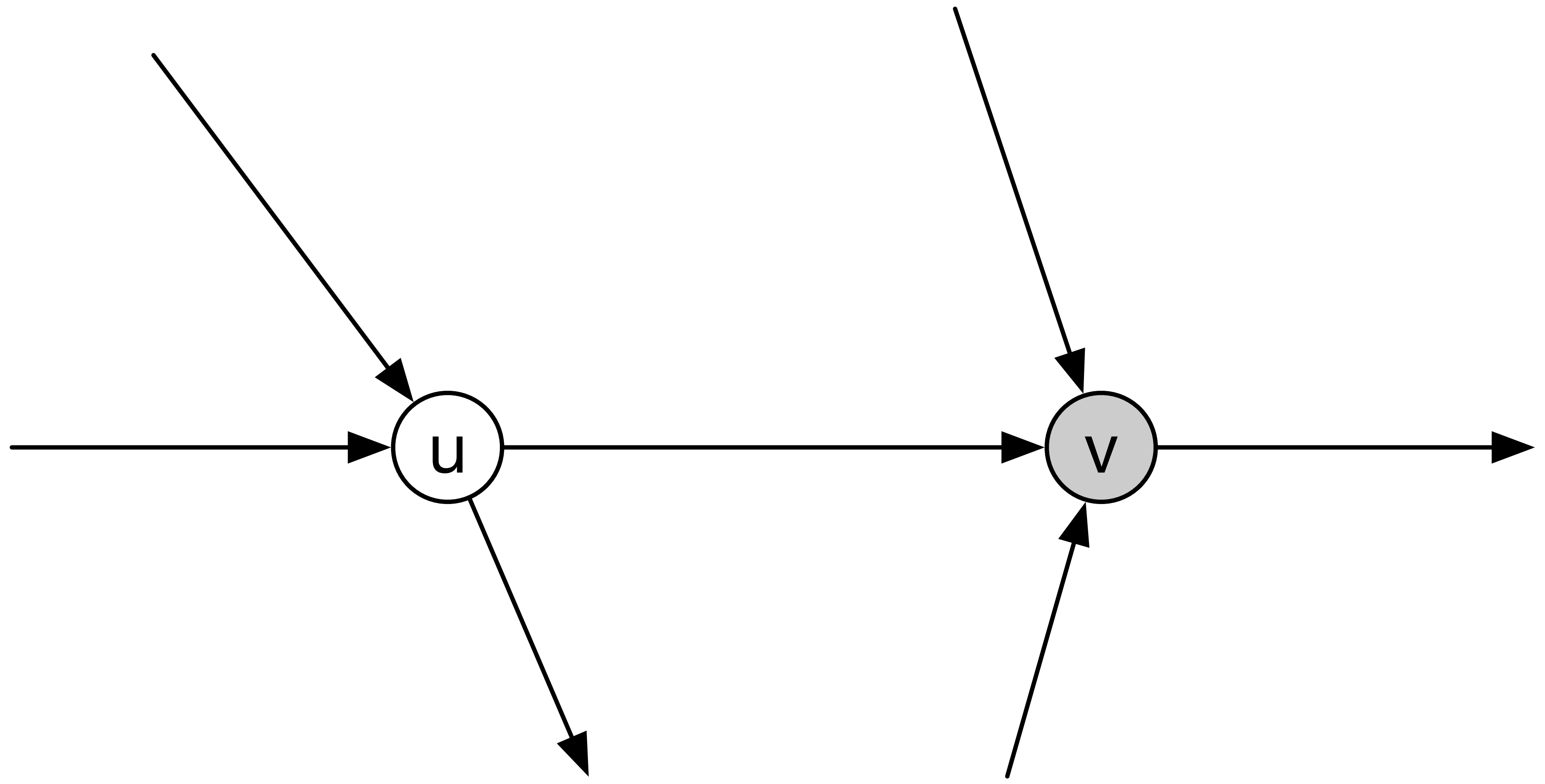


Edge Defective Edge Coloring



Semi Matching

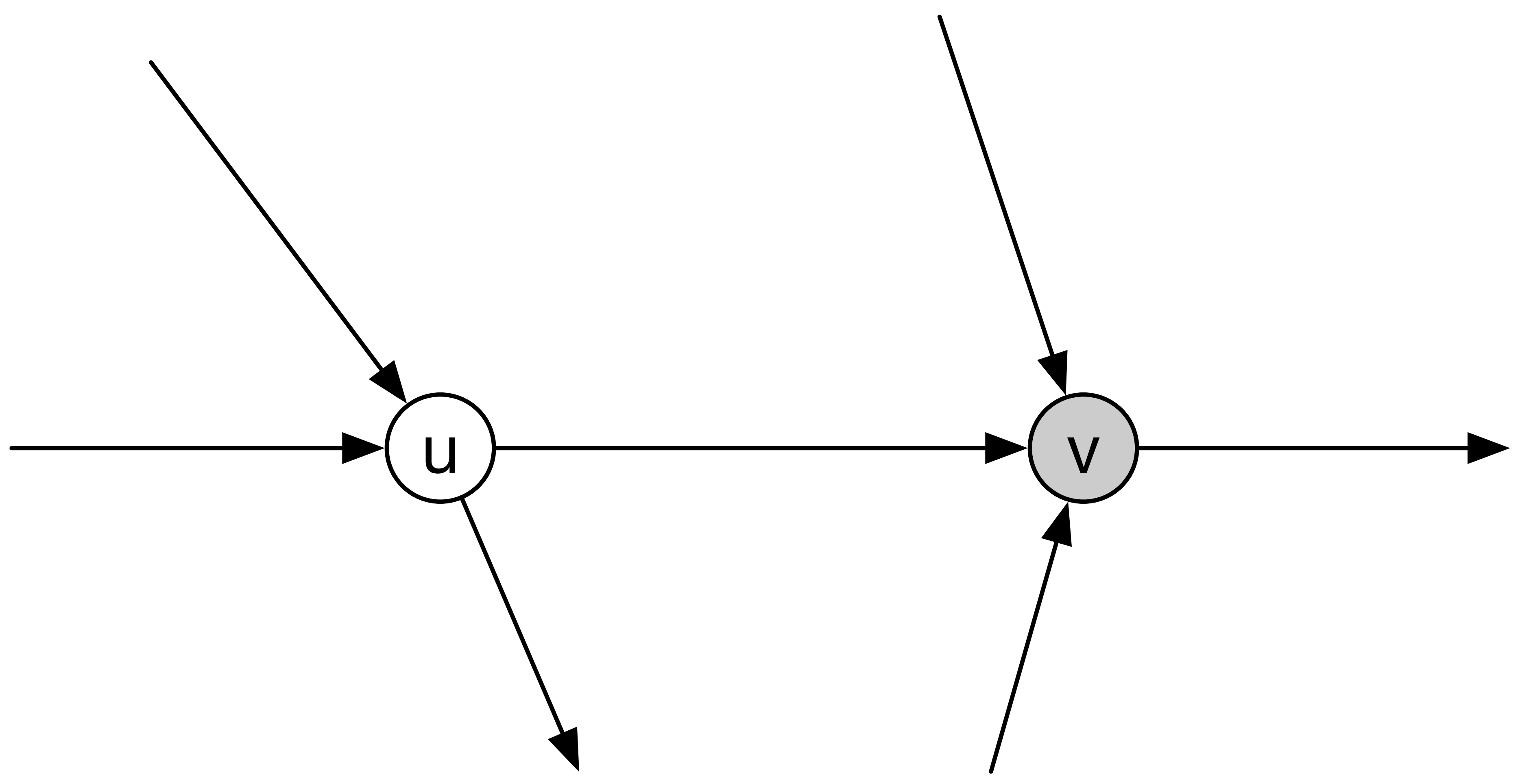
- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$



Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

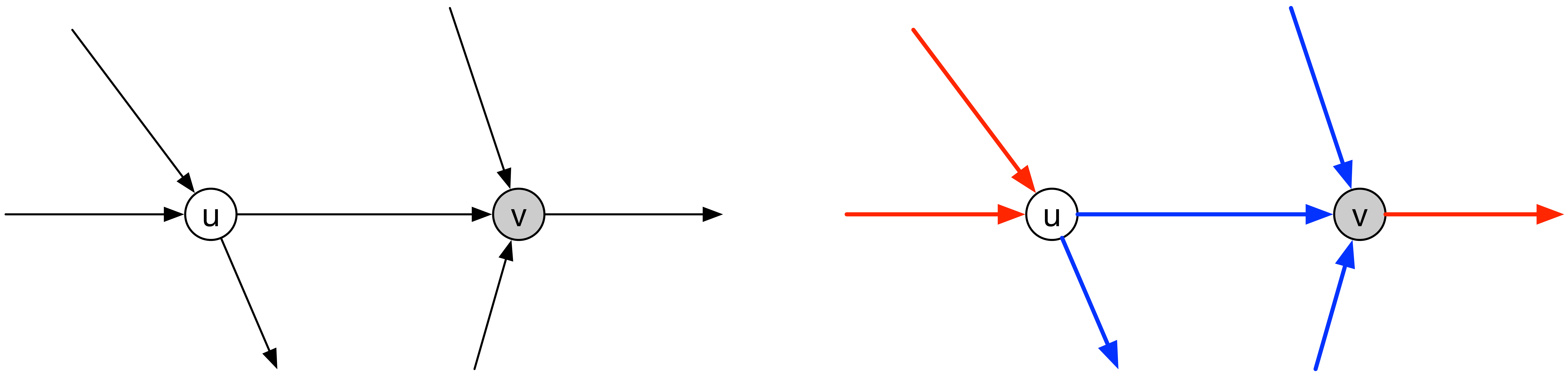
- Color blue the edges incoming on black nodes, color red the edges incoming on white nodes



Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

- Color blue the edges incoming on black nodes, color red the edges incoming on white nodes

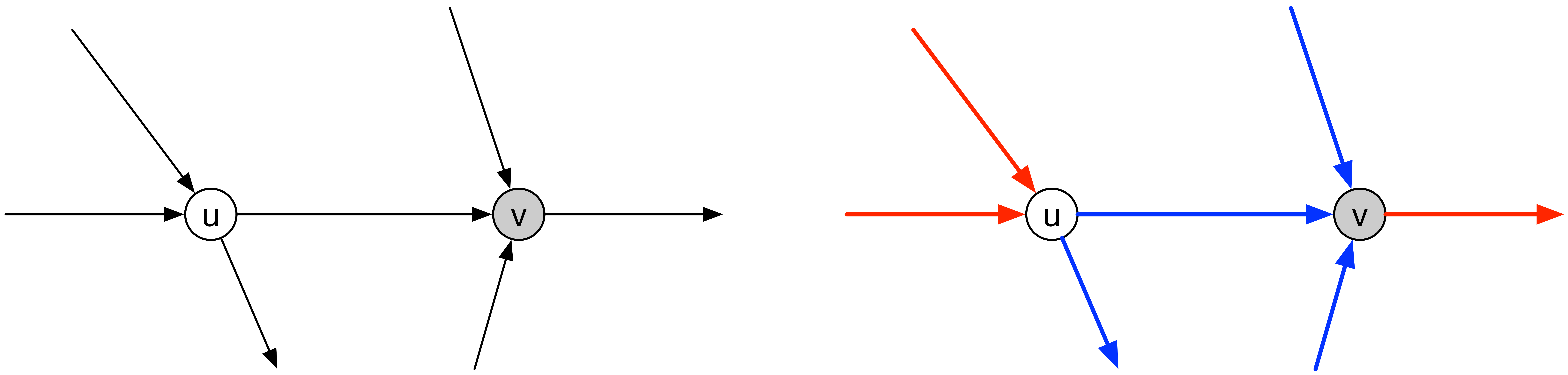


Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

- Color blue the edges incoming on black nodes, color red the edges incoming on white nodes

- Defect: $(\deg_{\text{in}}(v) - 1) + (\Delta - \deg_{\text{in}}(u) - 1) \leq$

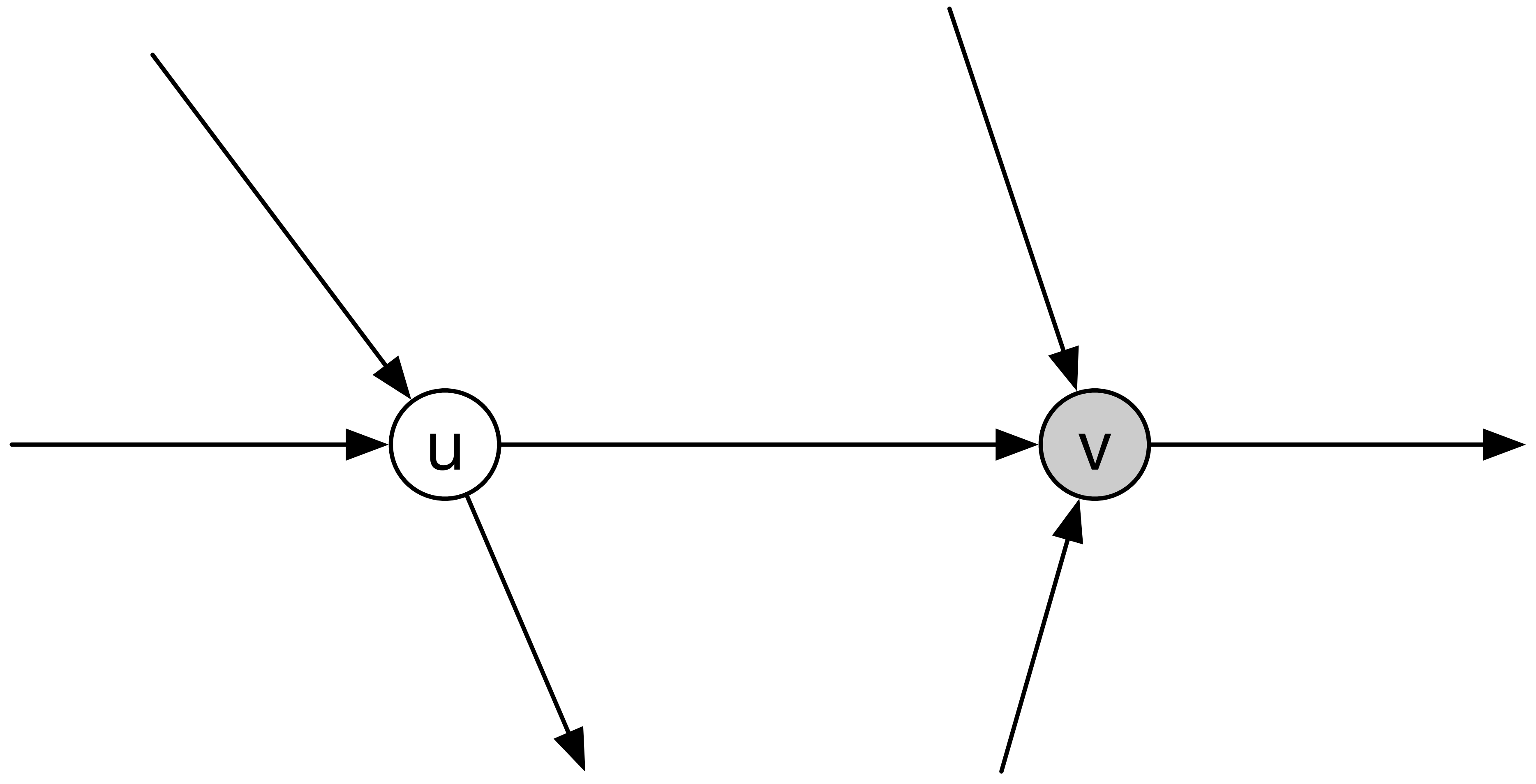


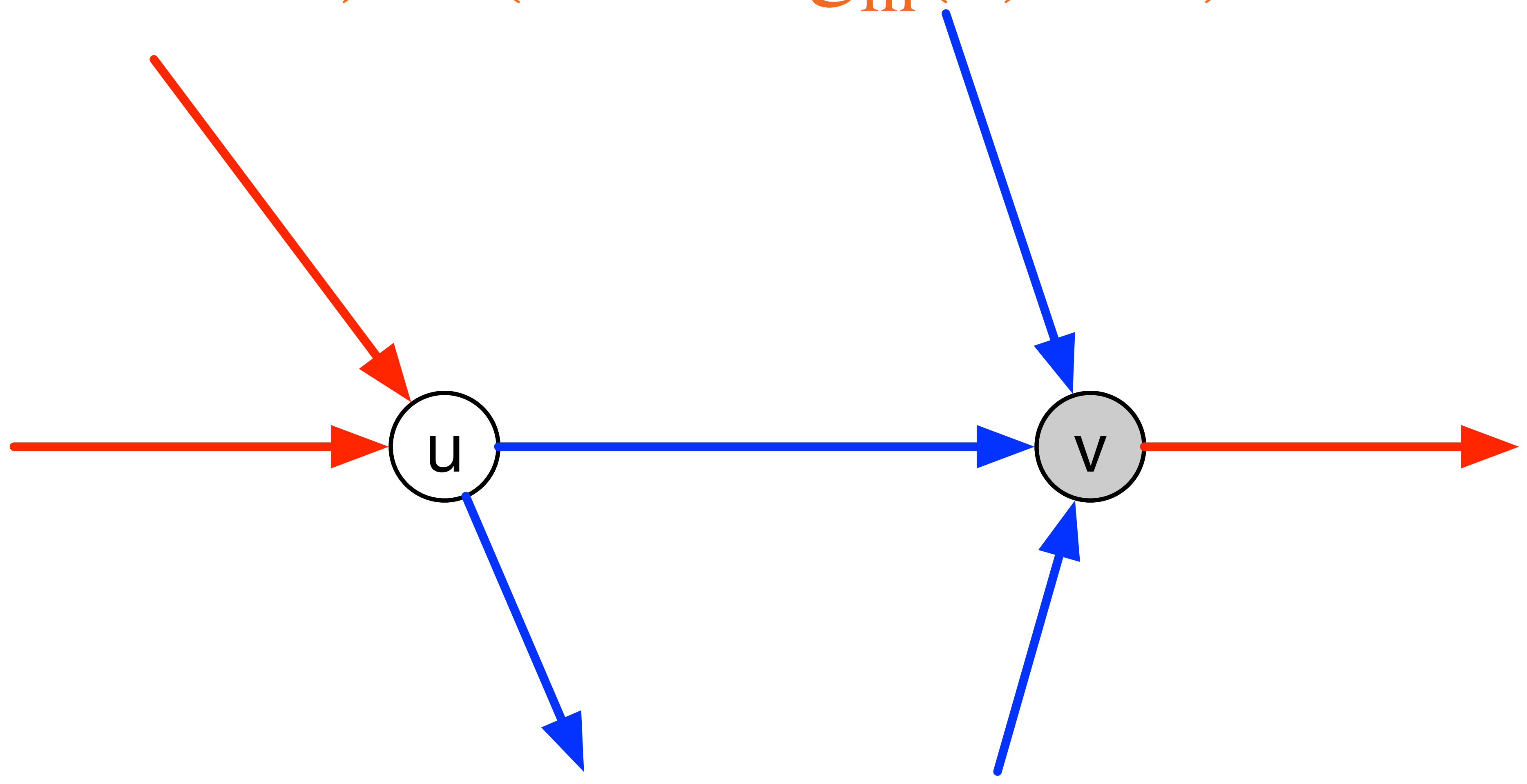
Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{in}(v) \leq \deg_{in}(u) + 1$

- Color blue the edges incoming on black nodes, color red the edges incoming on white nodes

- Defect: $(\deg_{in}(v) - 1) + (\Delta - \deg_{in}(u) - 1) \leq$
$(\deg_{in}(u) + 1 - 1) + (\Delta - \deg_{in}(u) - 1) = \Delta - 1$





Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

Efficient Load-Balancing through Distributed Token Dropping

[Brandt, Keller, Rybicki, Suomela, Uitto 2021]

This problem can be solved in $O(\Delta^4)$ rounds!

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$
  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:

  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

  - The conversion only works on **regular graphs, we do not have that**

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$
  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**
  - The conversion only works on **regular graphs, we do not have that**
  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$
  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**
  - The conversion only works on **regular graphs, we do not have that**

This is easy to handle

  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:

  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

  - The conversion only works on **regular graphs, we do not have that**

    This is easy to handle

  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

    This is very challenging

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$
  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

This is not hard

  - The conversion only works on **regular graphs, we do not have that**

This is easy to handle

  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

This is very challenging

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:

  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

    This is interesting

  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

    This is not hard

  - The conversion only works on **regular graphs, we do not have that**

    This is easy to handle

  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

    This is very challenging

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:

  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

    This is interesting

  - We can turn a semi-matching into an edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

    This is not hard

  - The conversion only works on **regular graphs, we do not have that**

    This is easy to handle

  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

    This is very challenging

Removing the 2-coloring requirement

Removing the 2-coloring requirement

- The $(\varepsilon\Delta + \lfloor \Delta/2 \rfloor)$-defective vertex **4-coloring** problem can be solved in $O(\log^* n)$ rounds [Barenboim, Elkin, Kuhn 2014]

Removing the 2-coloring requirement

- The $(\varepsilon\Delta + \lfloor \Delta/2 \rfloor)$-defective vertex **4-coloring** problem can be solved in $O(\log^* n)$ rounds [Barenboim, Elkin, Kuhn 2014]

- We can color the nodes of a graph with **4 colors**, such that, for each node, **not much more than half** of its neighbors have the same color

Removing the 2-coloring requirement

- The $(\varepsilon\Delta + \lfloor \Delta/2 \rfloor)$-defective vertex **4-coloring** problem can be solved in $O(\log^* n)$ rounds [Barenboim, Elkin, Kuhn 2014]

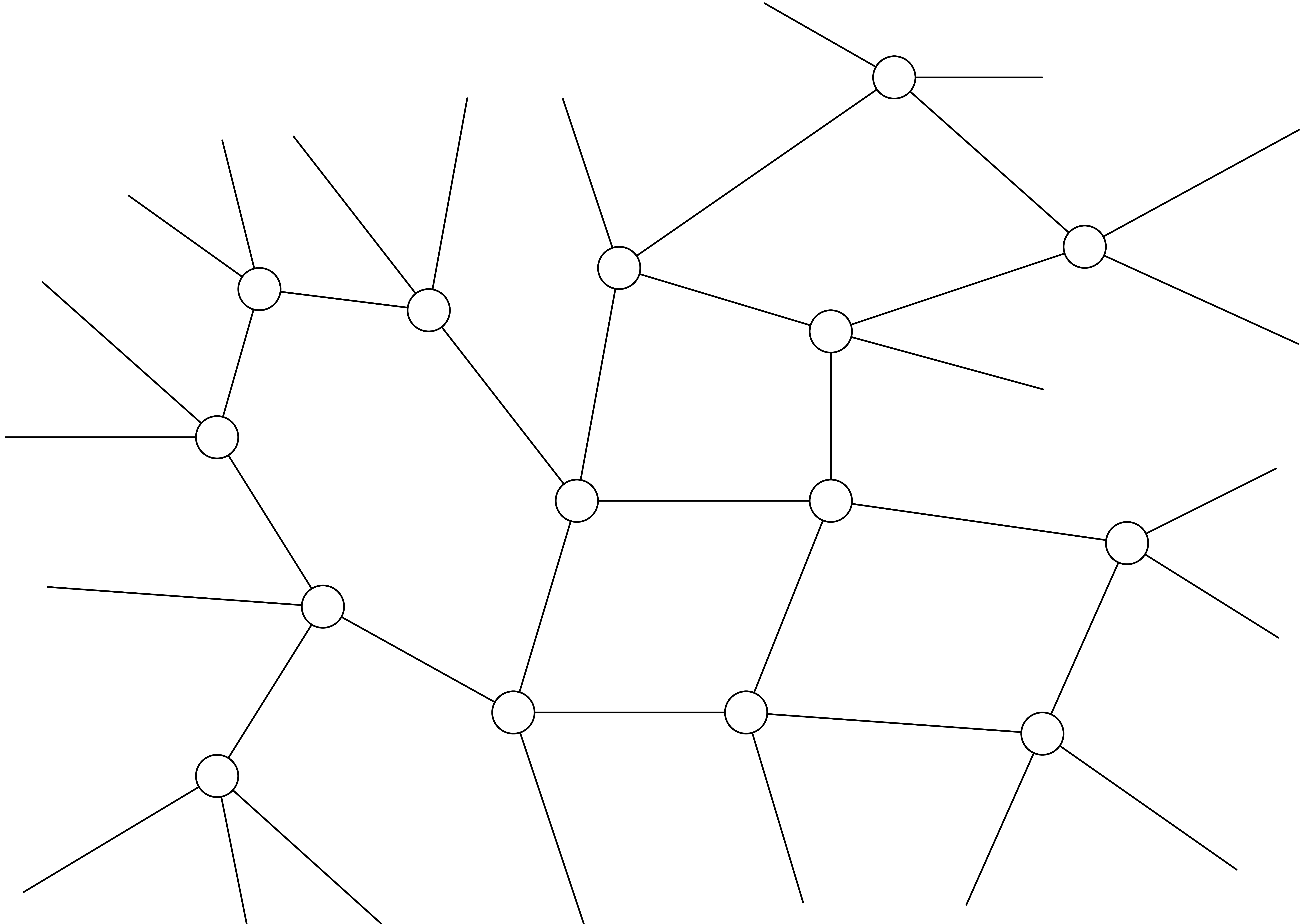
- We can color the nodes of a graph with **4 colors**, such that, for each node, **not much more than half** of its neighbors have the same color

- Let's **ignore** monochromatic edges, we can **recurse** on them later

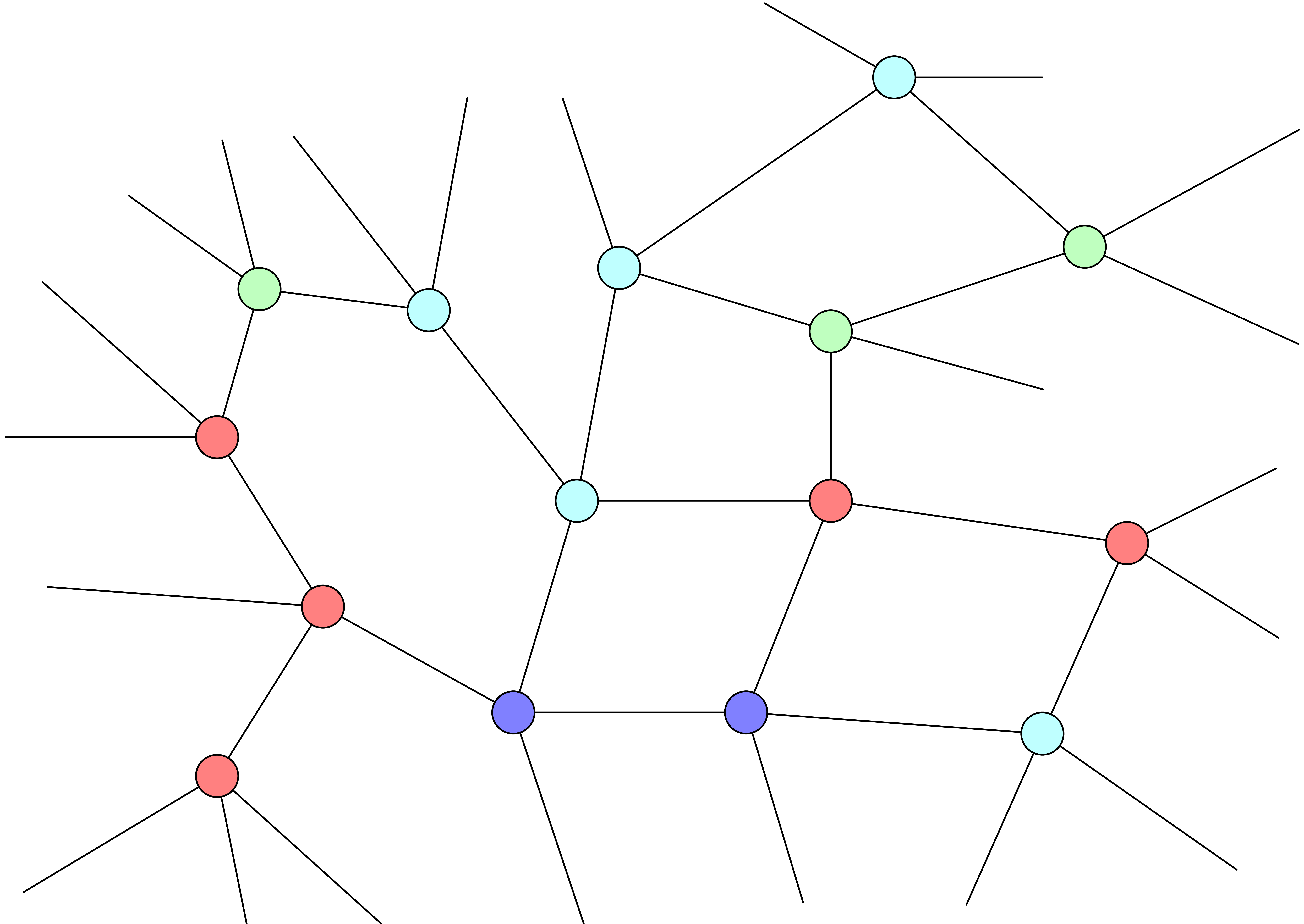
Removing the 2-coloring requirement

- The $(\varepsilon\Delta + \lfloor \Delta/2 \rfloor)$-defective vertex **4-coloring** problem can be solved in $O(\log^* n)$ rounds [Barenboim, Elkin, Kuhn 2014]
- We can color the nodes of a graph with **4 colors**, such that, for each node, **not much more than half** of its neighbors have the same color
- Let's **ignore** monochromatic edges, we can **recurse** on them later
- We have a **4** coloring, we need a **2** coloring...

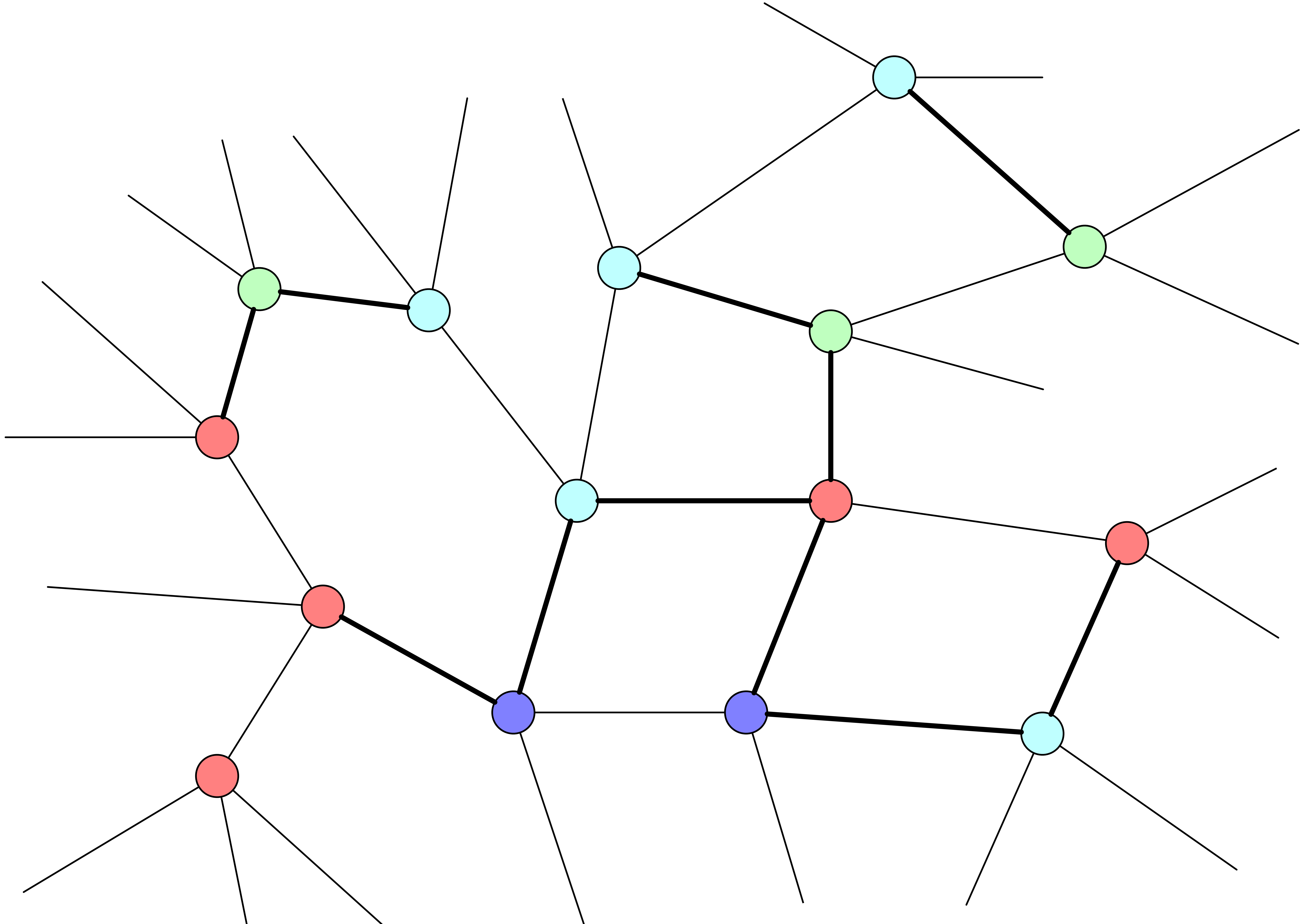
Removing the 2-coloring requirement



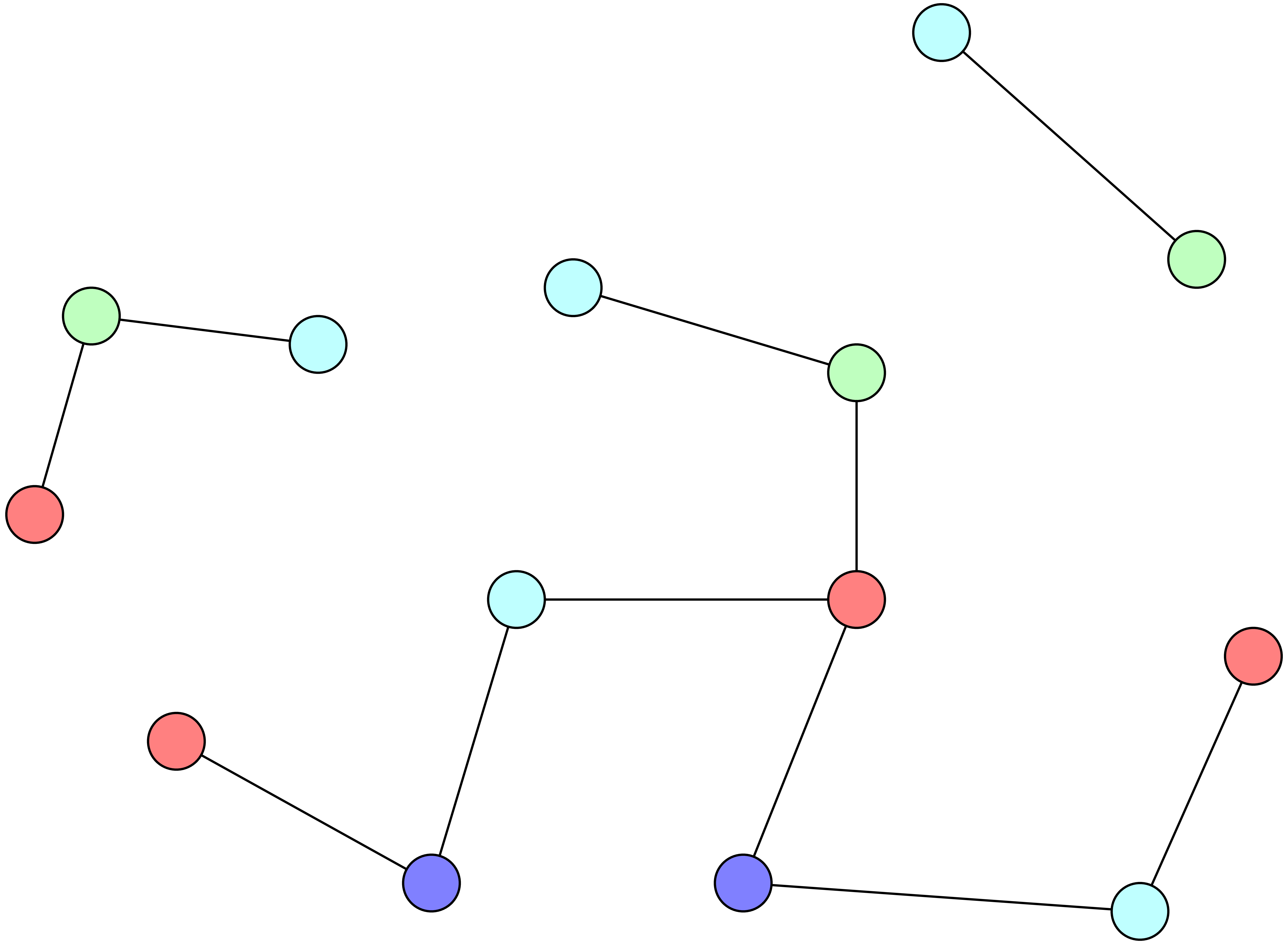
Removing the 2-coloring requirement



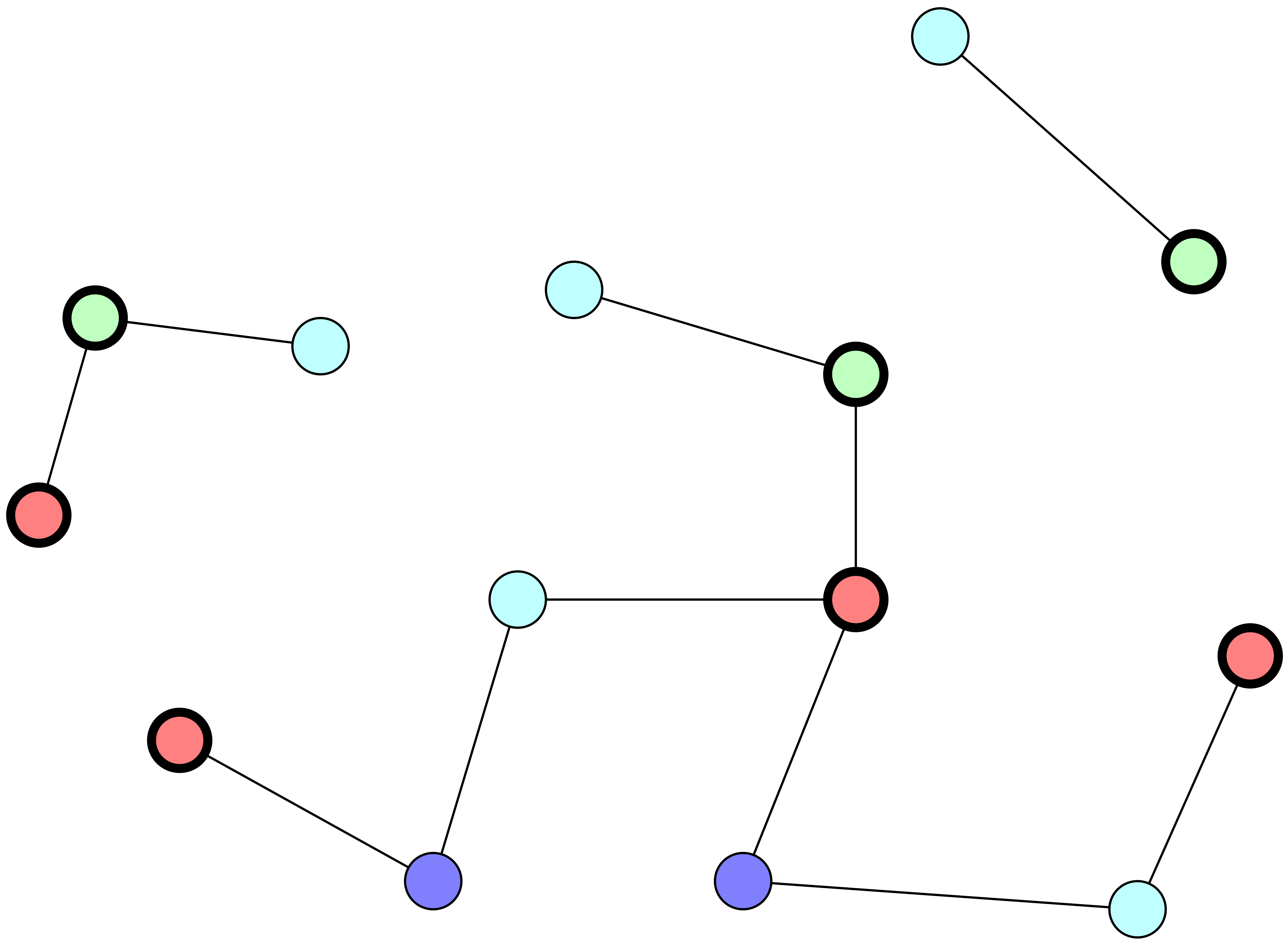
Removing the 2-coloring requirement



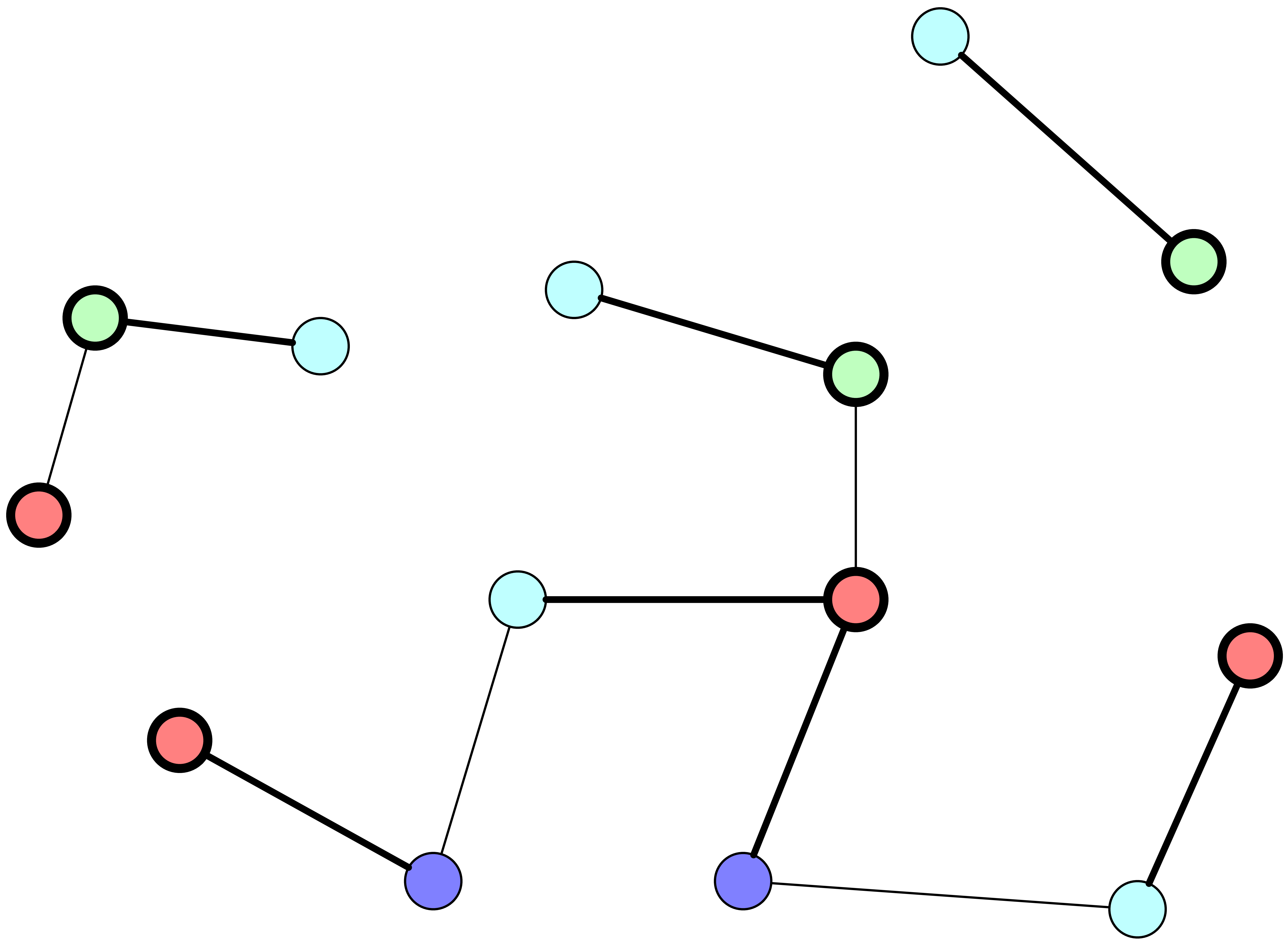
Removing the 2-coloring requirement



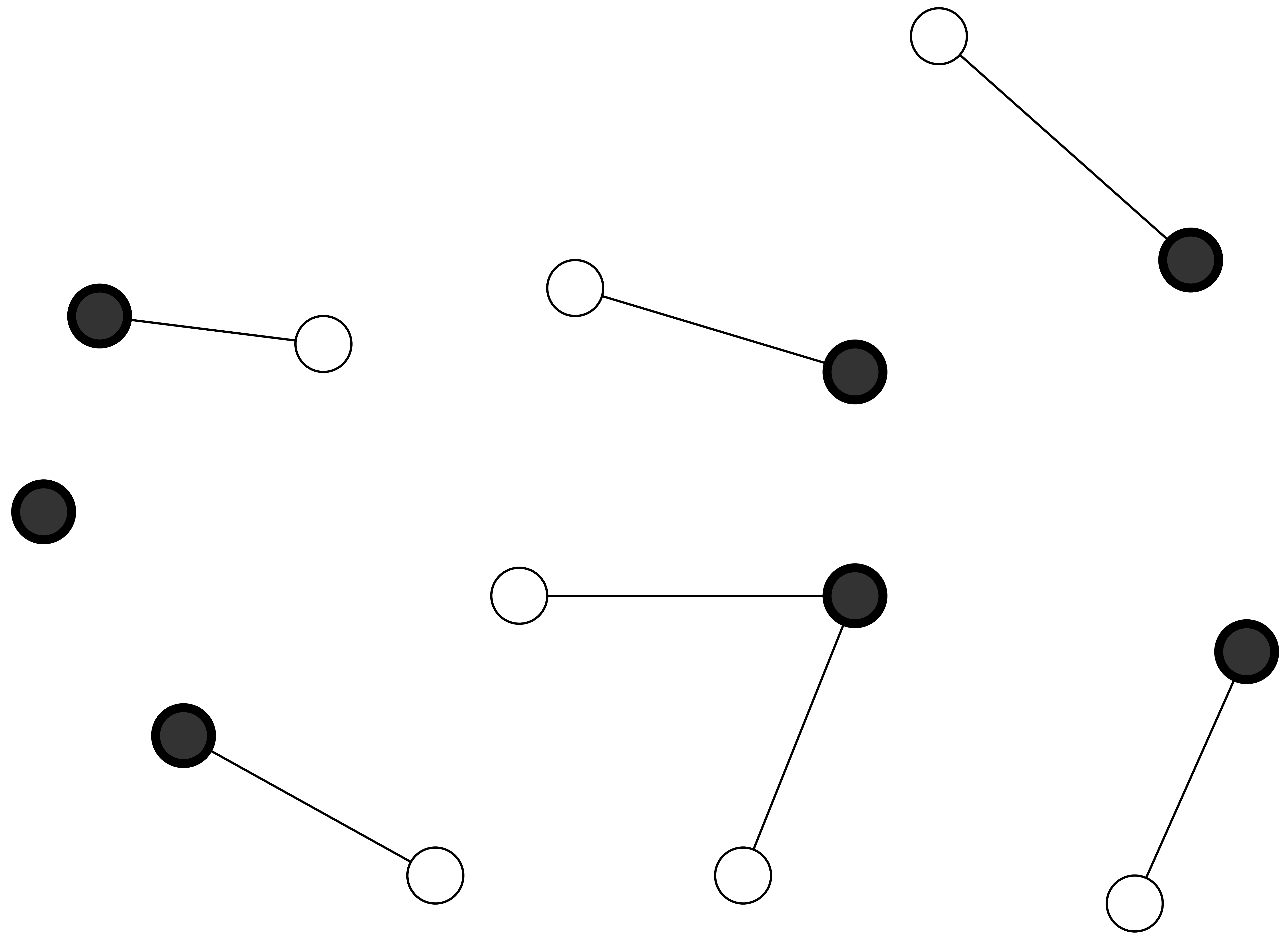
Removing the 2-coloring requirement



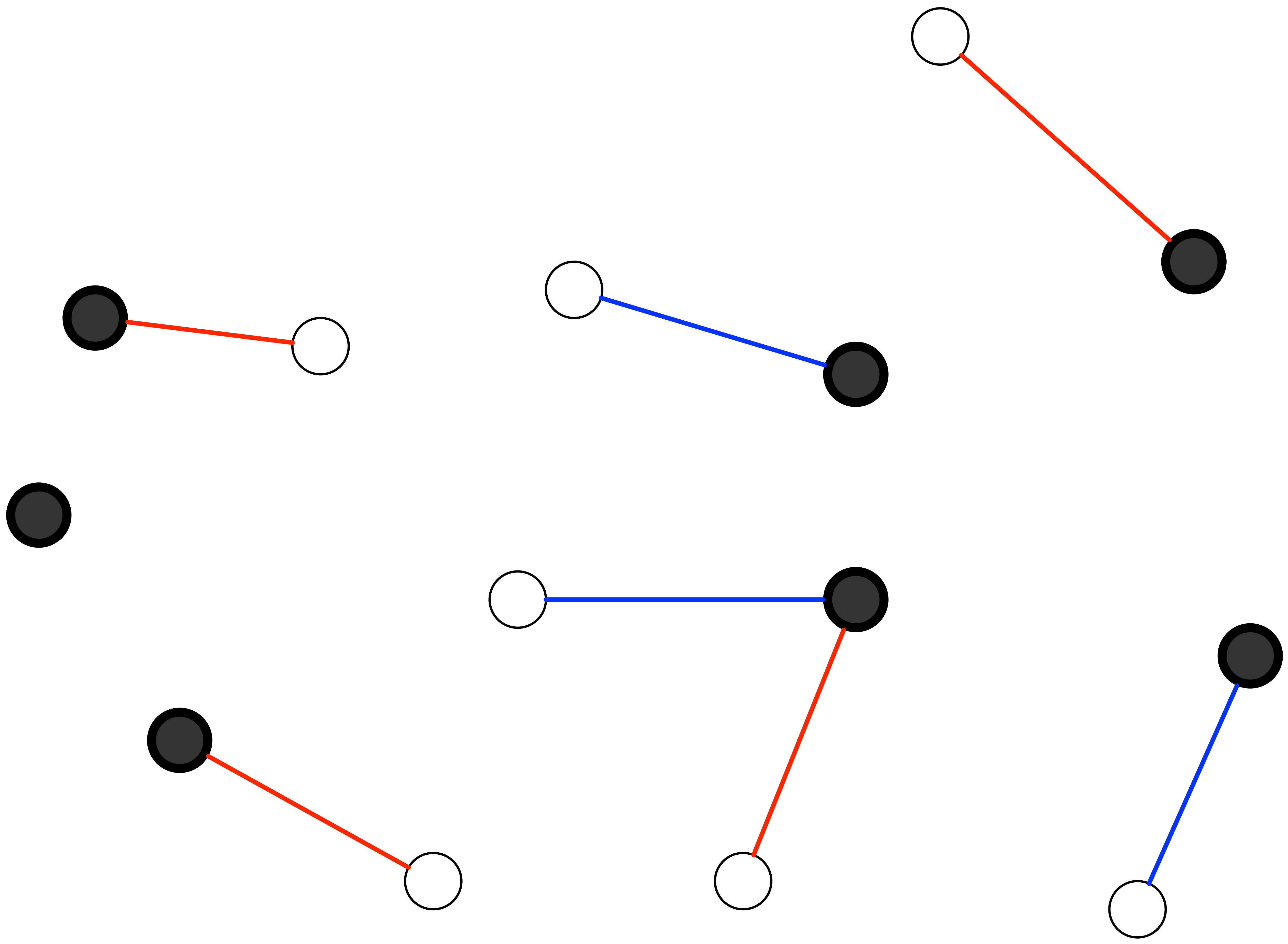
Removing the 2-coloring requirement



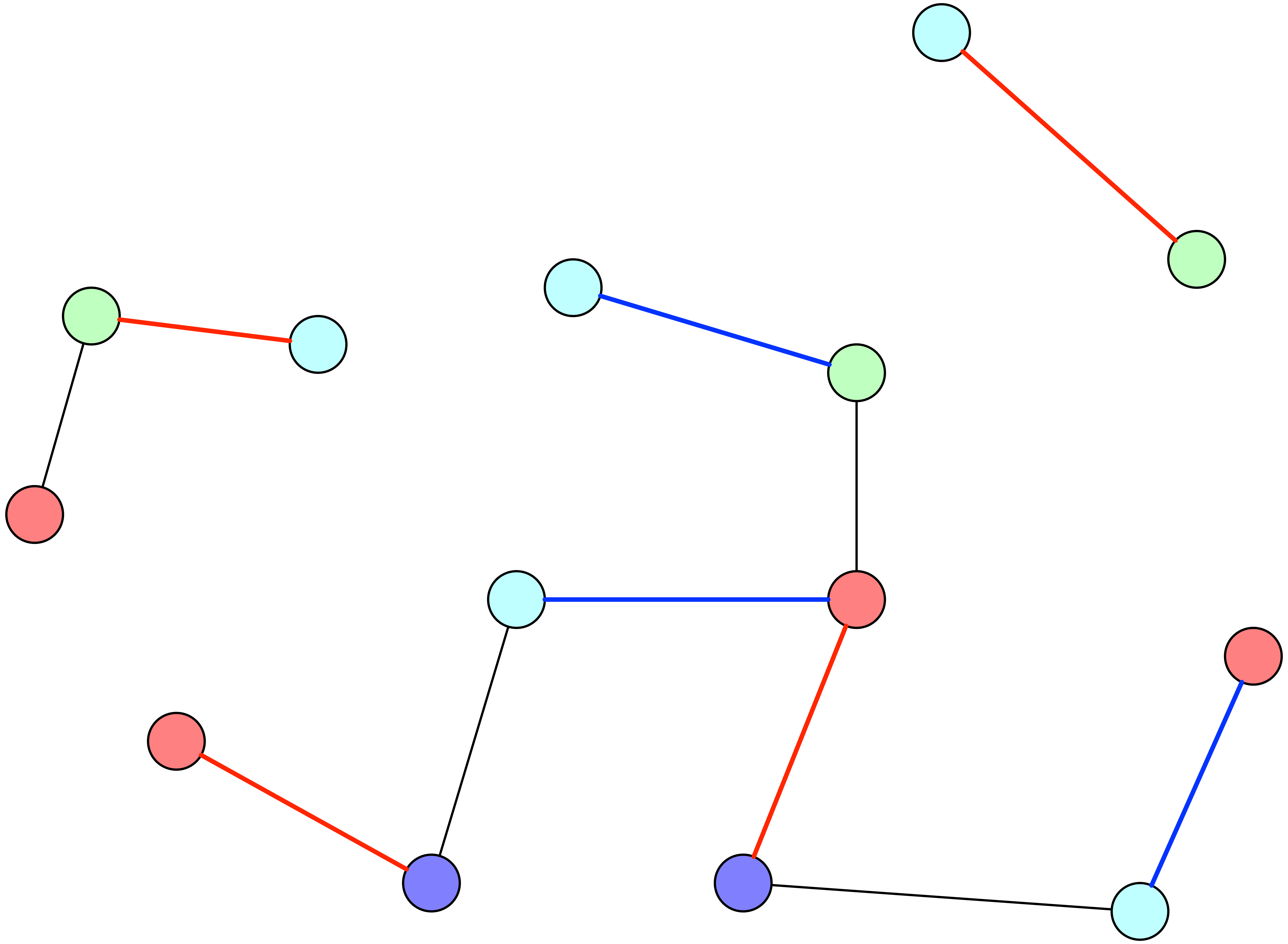
Removing the 2-coloring requirement



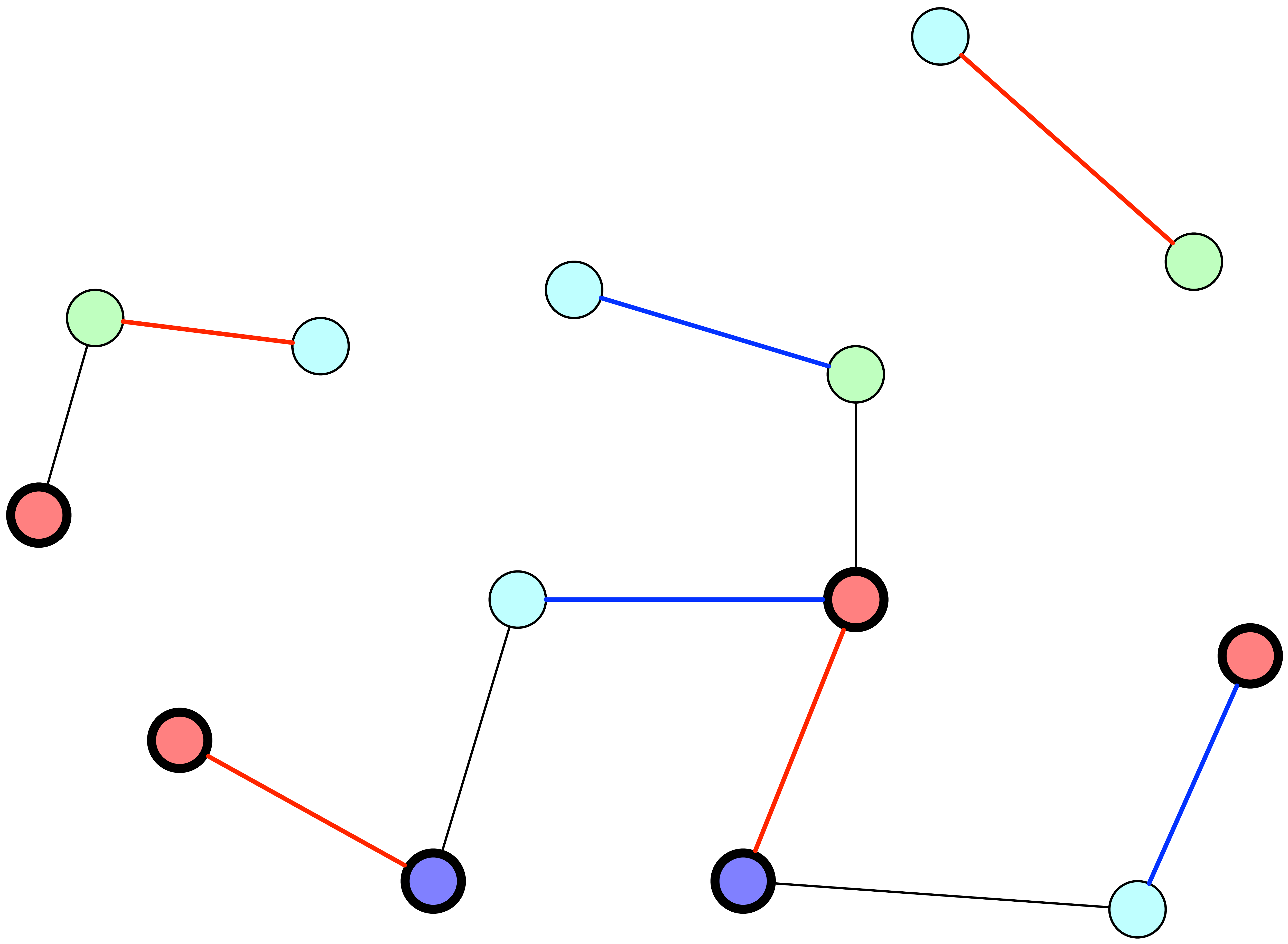
Removing the 2-coloring requirement



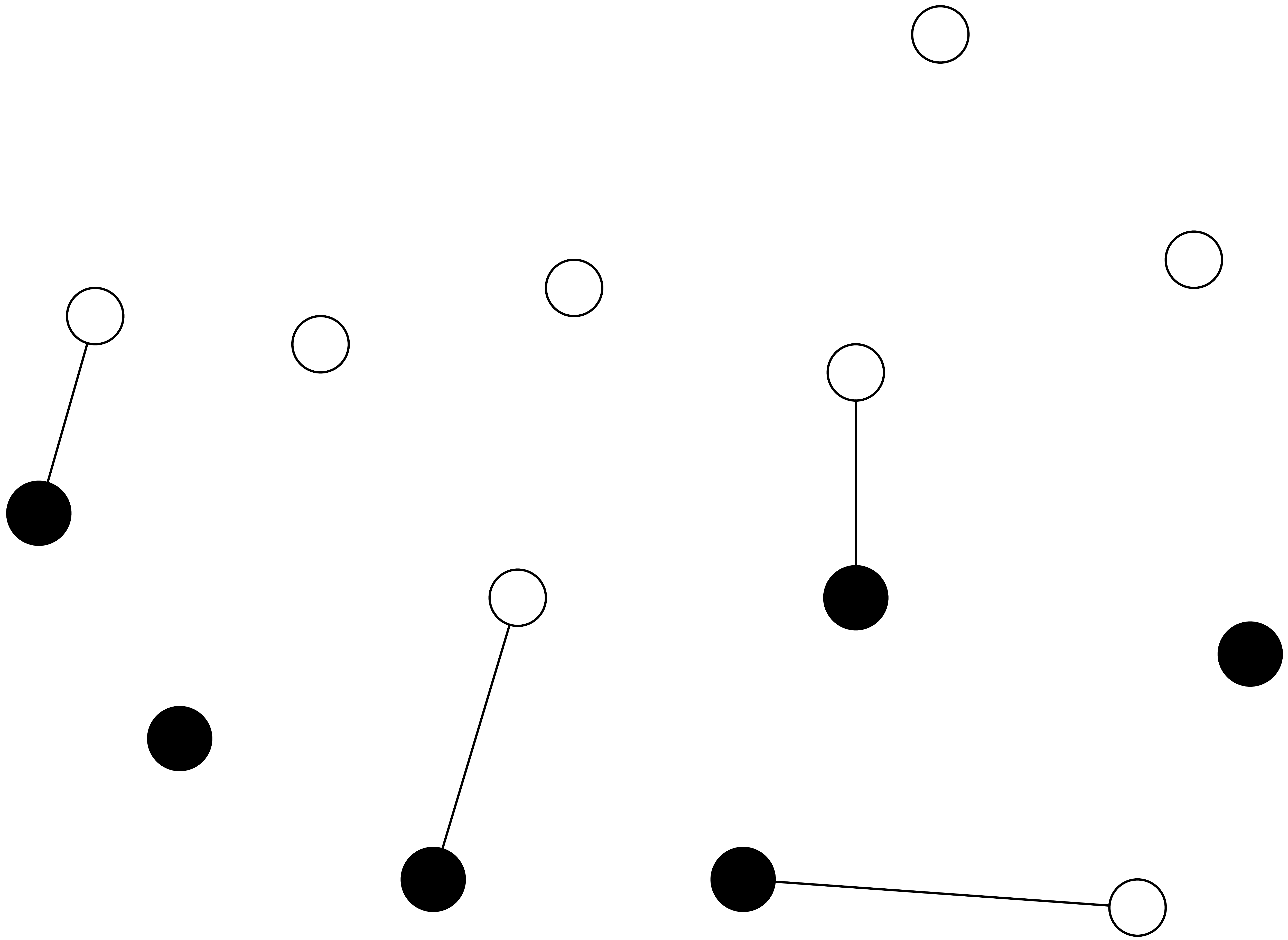
Removing the 2-coloring requirement



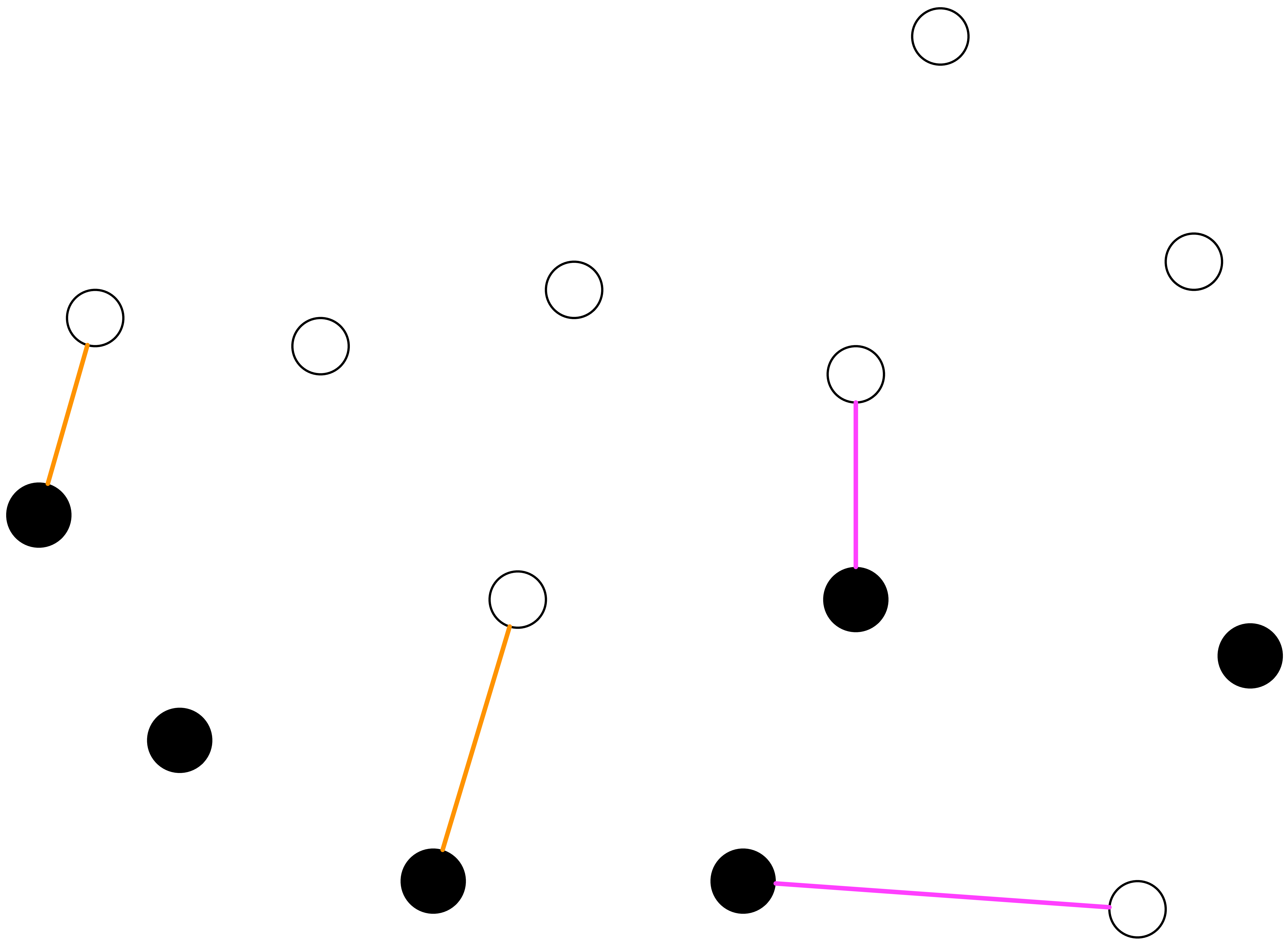
Removing the 2-coloring requirement



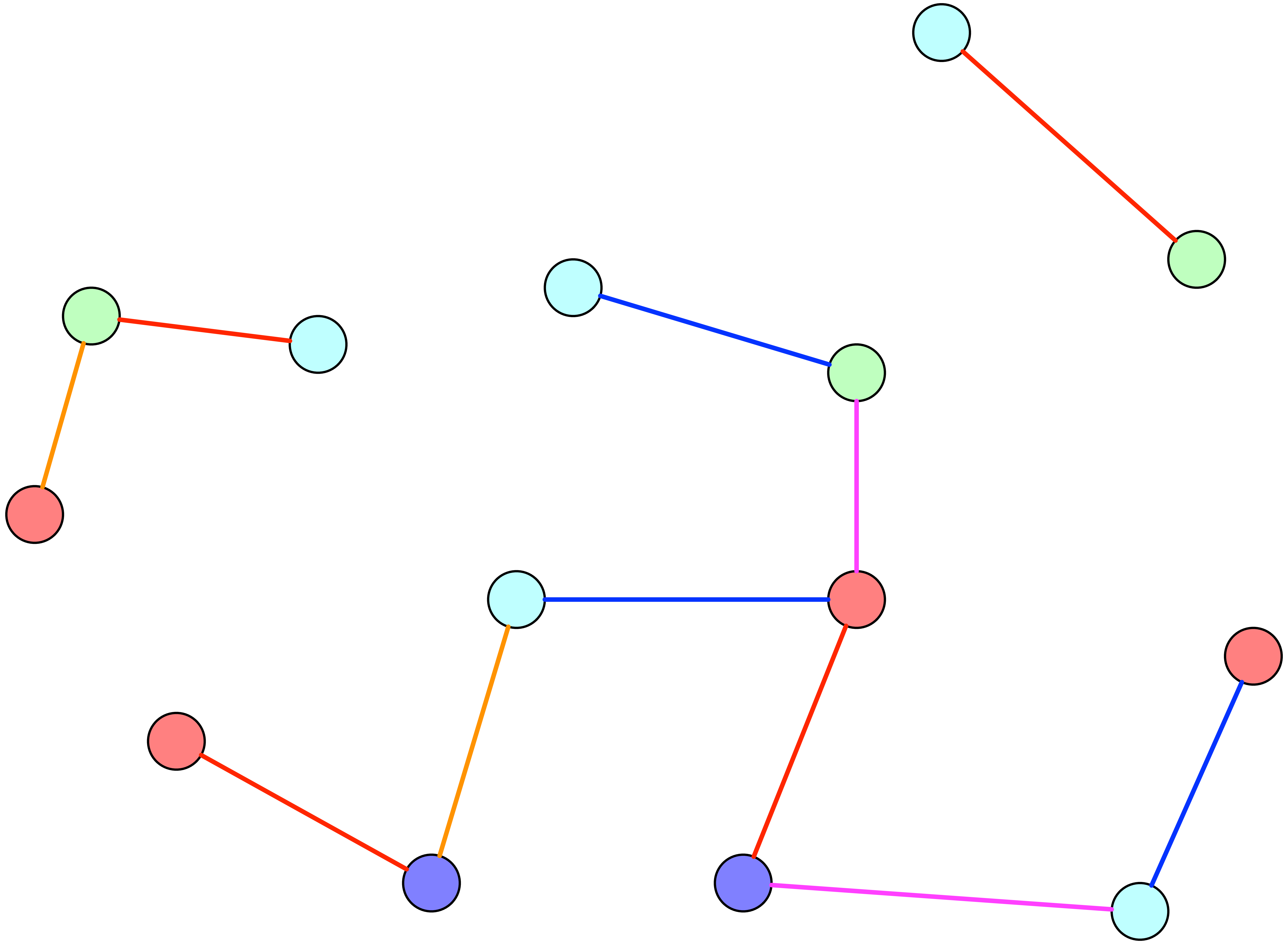
Removing the 2-coloring requirement



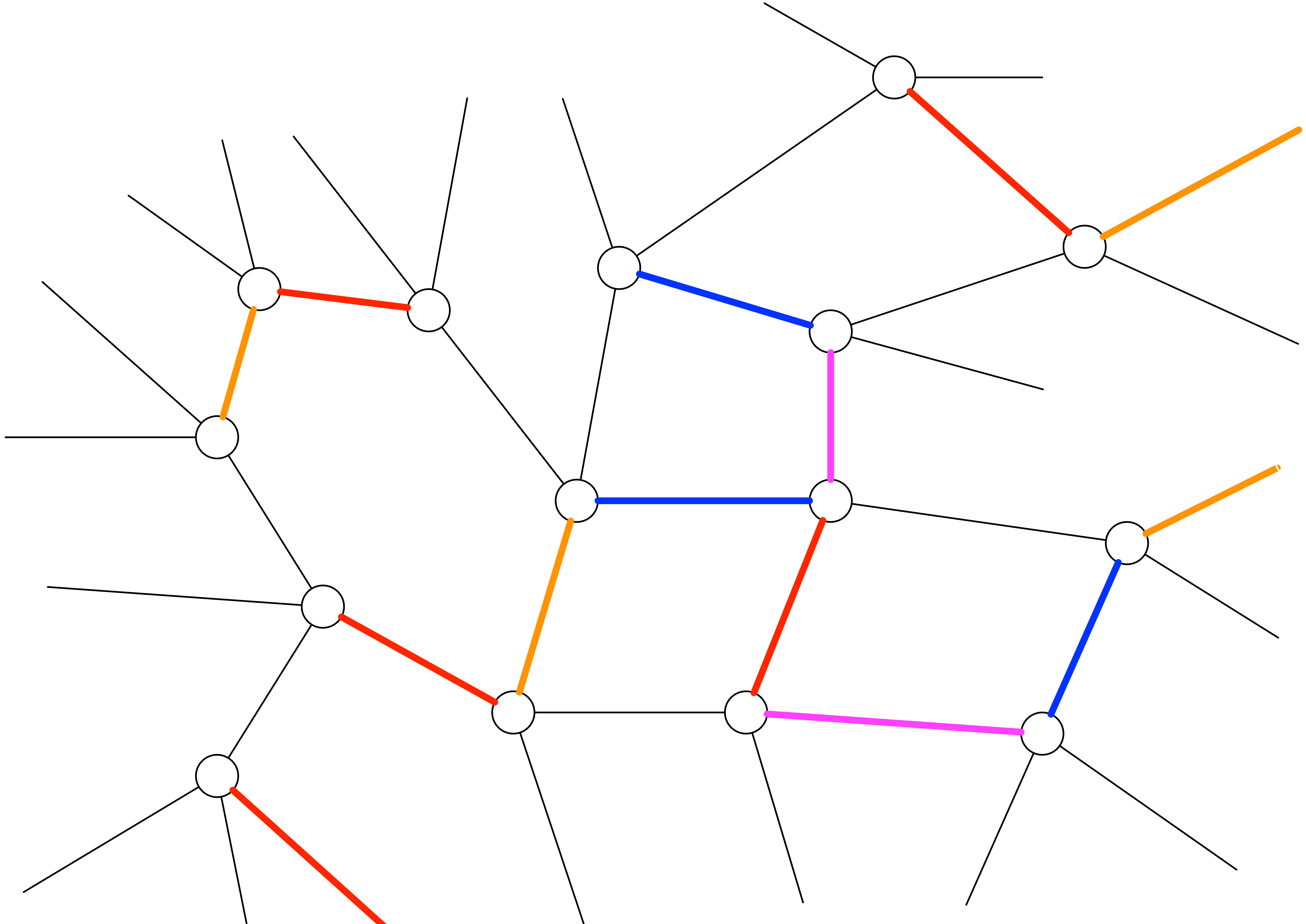
Removing the 2-coloring requirement



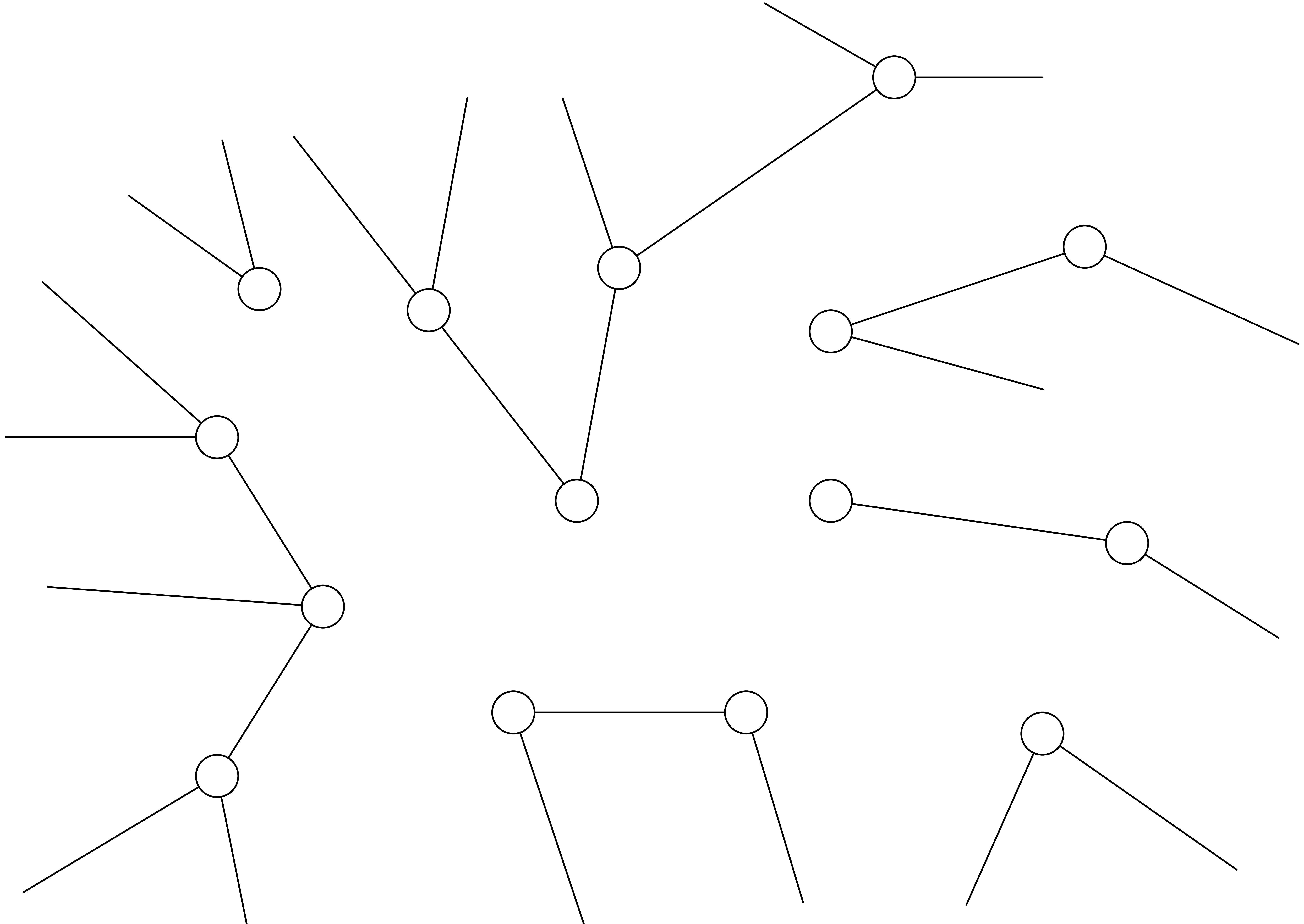
Removing the 2-coloring requirement



Removing the 2-coloring requirement



Removing the 2-coloring requirement



Removing the 2-coloring requirement

Removing the 2-coloring requirement

- Assume we have an $O(\Delta)$-edge coloring algorithm for **2-vertex colored graphs**.

Removing the 2-coloring requirement

- Assume we have an $O(\Delta)$-edge coloring algorithm for **2-vertex colored graphs**.
- We can use $c\Delta$ colors to **partially edge-color the graph**, such that the **maximum vertex degree roughly halves**

Removing the 2-coloring requirement

- Assume we have an $O(\Delta)$-edge coloring algorithm for **2-vertex colored graphs**.
- We can use $c\Delta$ colors to **partially edge-color the graph**, such that the **maximum vertex degree roughly halves**
- We can then use $c\Delta/2$ other colors to partially color the remaining edges, such that the maximum vertex degree roughly halves again...

Removing the 2-coloring requirement

- Assume we have an $O(\Delta)$-edge coloring algorithm for **2-vertex colored graphs**.
- We can use $c\Delta$ colors to **partially edge-color the graph**, such that the **maximum vertex degree roughly halves**
- We can then use $c\Delta/2$ other colors to partially color the remaining edges, such that the maximum vertex degree roughly halves again...
- Repeat... at step **i**, we use $c\Delta/2^i$ colors, and the **maximum degree goes down by a factor 2**

Removing the 2-coloring requirement

- Assume we have an $O(\Delta)$-edge coloring algorithm for **2-vertex colored graphs**.
- We can use $c\Delta$ colors to **partially edge-color the graph**, such that the **maximum vertex degree roughly halves**
- We can then use $c\Delta/2$ other colors to partially color the remaining edges, such that the maximum vertex degree roughly halves again...
- Repeat... at step **i**, we use $c\Delta/2^i$ colors, and the **maximum degree goes down by a factor 2**
- With $2c\Delta = O(\Delta)$ colors we color all the edges

Removing the 2-coloring requirement

- Assume we have an $O(\Delta)$-edge coloring algorithm for **2-vertex colored graphs**.
- We can use $c\Delta$ colors to **partially edge-color the graph**, such that the **maximum vertex degree roughly halves**
- We can then use $c\Delta/2$ other colors to partially color the remaining edges, such that the maximum vertex degree roughly halves again...
- Repeat... at step **i**, we use $c\Delta/2^i$ colors, and the **maximum degree goes down by a factor 2**
- With $2c\Delta = O(\Delta)$ colors we color all the edges

Given a T round algorithm for $O(\Delta)$-edge coloring in bipartite 2-colored graphs, we can construct an algorithm for $O(\Delta)$-edge coloring in general graphs that runs in $O(T \log \Delta + \log^* n)$ rounds

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

    This is interesting

  - We can turn a semi-matching into a edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

    This is not hard

  - The conversion only works on **regular graphs, we do not have that**

    This is easy to handle

  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

    This is very challenging

Issues

- **Semi-matching** solves **"balanced" edge 2-coloring**, but:
  - The running time is $O(\Delta^4)$, we want $O(\log^c \Delta)$

    This is interesting
  - We can turn a semi-matching into a edge 2-coloring **only if a 2-vertex coloring is given**, **we do not have that**

    This is not hard
  - The conversion only works on **regular graphs, we do not have that**

    This is easy to handle
  - The recursion schema solves $O(\Delta)$**-edge coloring,** not $(2\Delta - 1)$**-edge coloring.** For a better result, we need to solve a harder variant **(list coloring)**

    This is very challenging

Relaxed Semi Matching

Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$
- This allows us to reduce the maximum edge degree from $(2\Delta - 2)$ to $(\Delta - 1)$, i.e., the new edge degree is **1/2** the original edge degree. Unfortunately, $O(\Delta^4)$ is too expensive.

Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

- This allows us to reduce the maximum edge degree from $(2\Delta - 2)$ to $(\Delta - 1)$, i.e., the new edge degree is **1/2** the original edge degree. Unfortunately, $O(\Delta^4)$ is too expensive.

- How about $\dfrac{\left(1 + \frac{1}{\log \Delta}\right)}{2}$?

Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + 1$

- This allows us to reduce the maximum edge degree from $(2\Delta - 2)$ to $(\Delta - 1)$, i.e., the new edge degree is **1/2** the original edge degree. Unfortunately, $O(\Delta^4)$ is too expensive.

- How about $\dfrac{\left(1 + \frac{1}{\log \Delta}\right)}{2}$?

- In order to achieve this, it turns out that it is enough to solve a more relaxed variant of semi-matching, that satisfies $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + \dfrac{\Delta}{\log \Delta}$

Relaxed Semi Matching

Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + k$

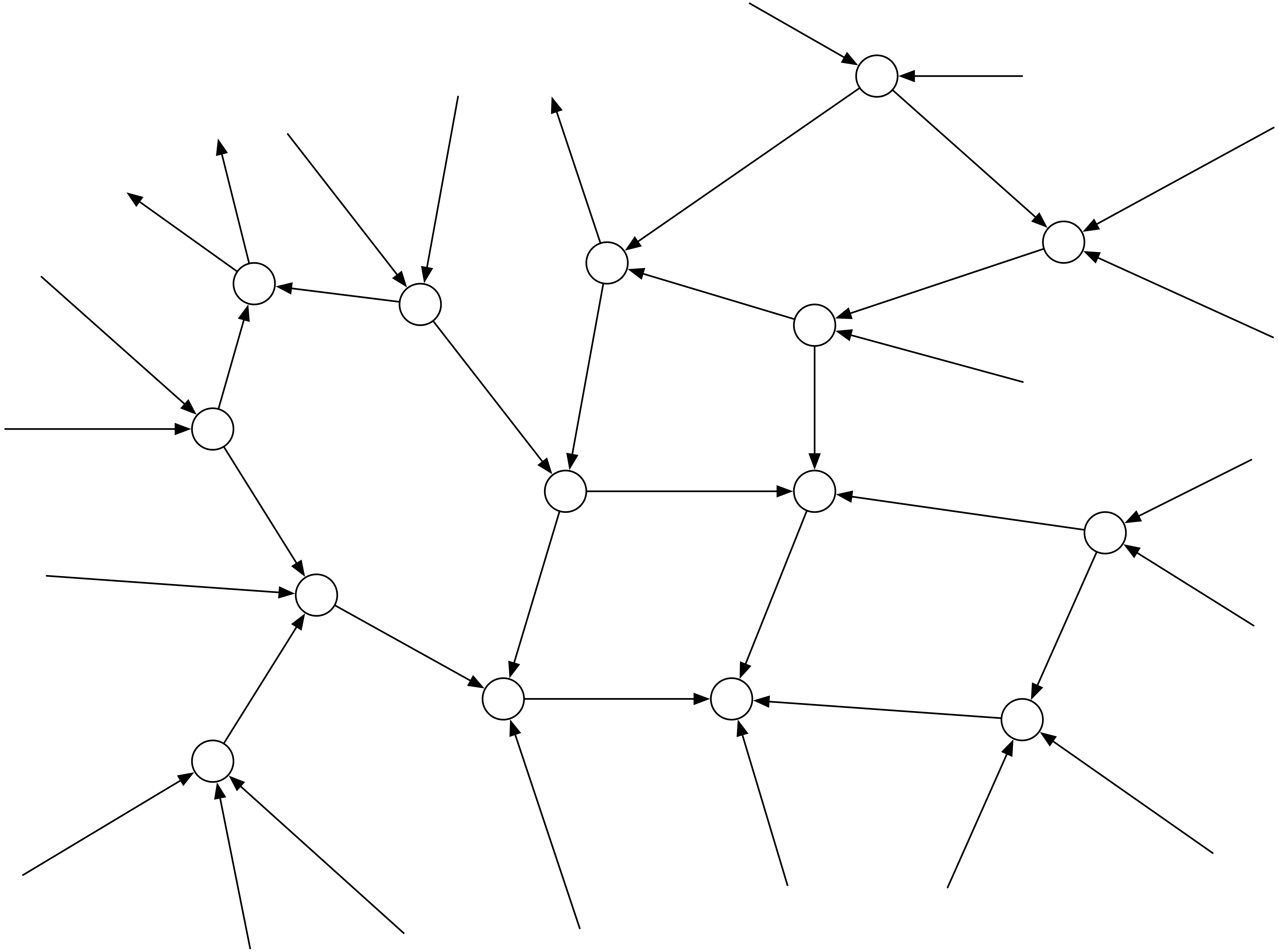
Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + k$
- This problem can be solved in $O(\Delta^5/k^5)$ rounds!

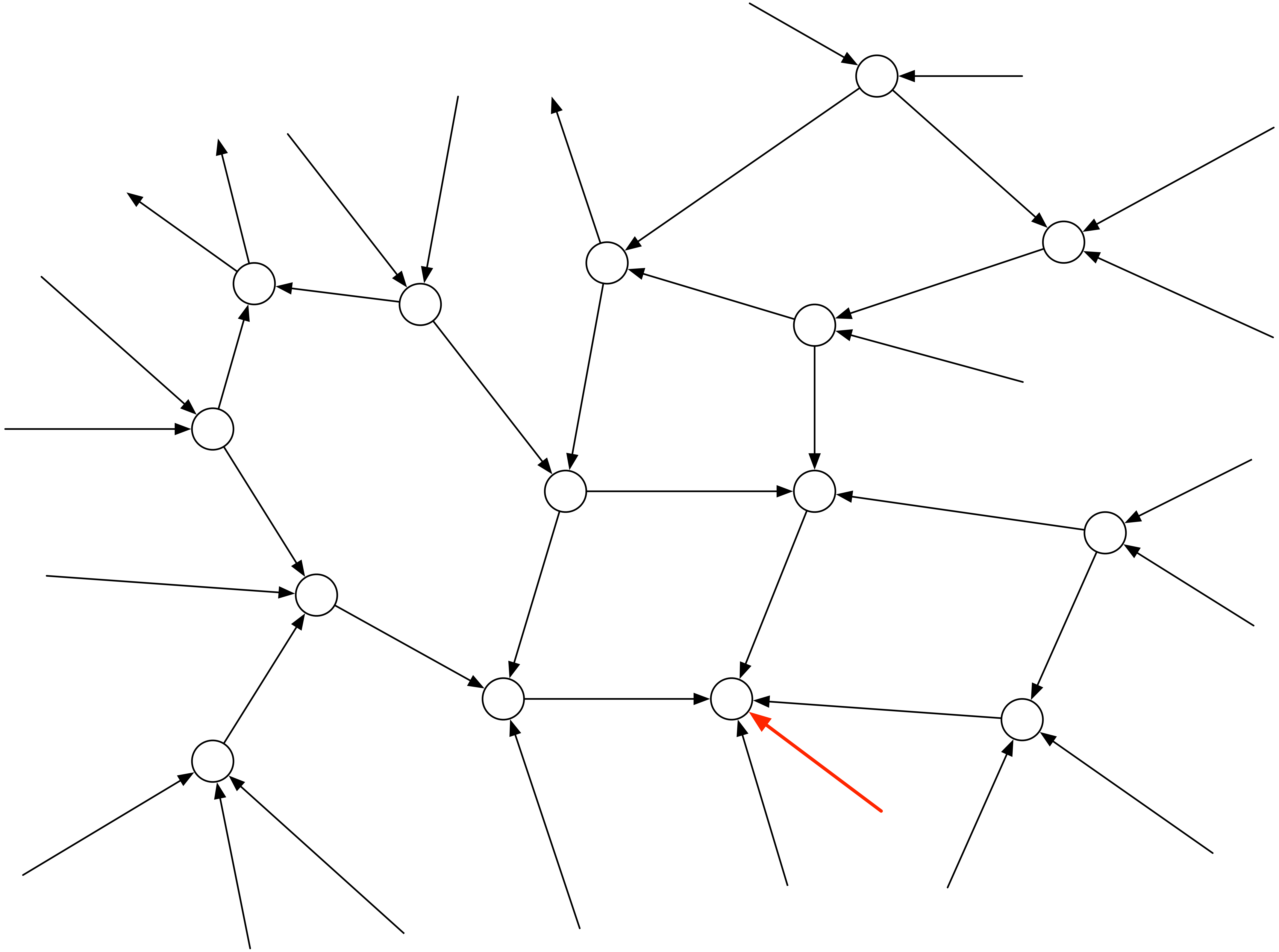
Relaxed Semi Matching

- **Orient** the edges of a graph such that, for each edge (u, v) oriented from u to v, it holds that $\deg_{\text{in}}(v) \leq \deg_{\text{in}}(u) + k$
- This problem can be solved in $O(\Delta^5/k^5)$ rounds!
- For $k = \dfrac{\Delta}{\log \Delta}$, this gives an $O(\log^5 \Delta)$ round algorithm!

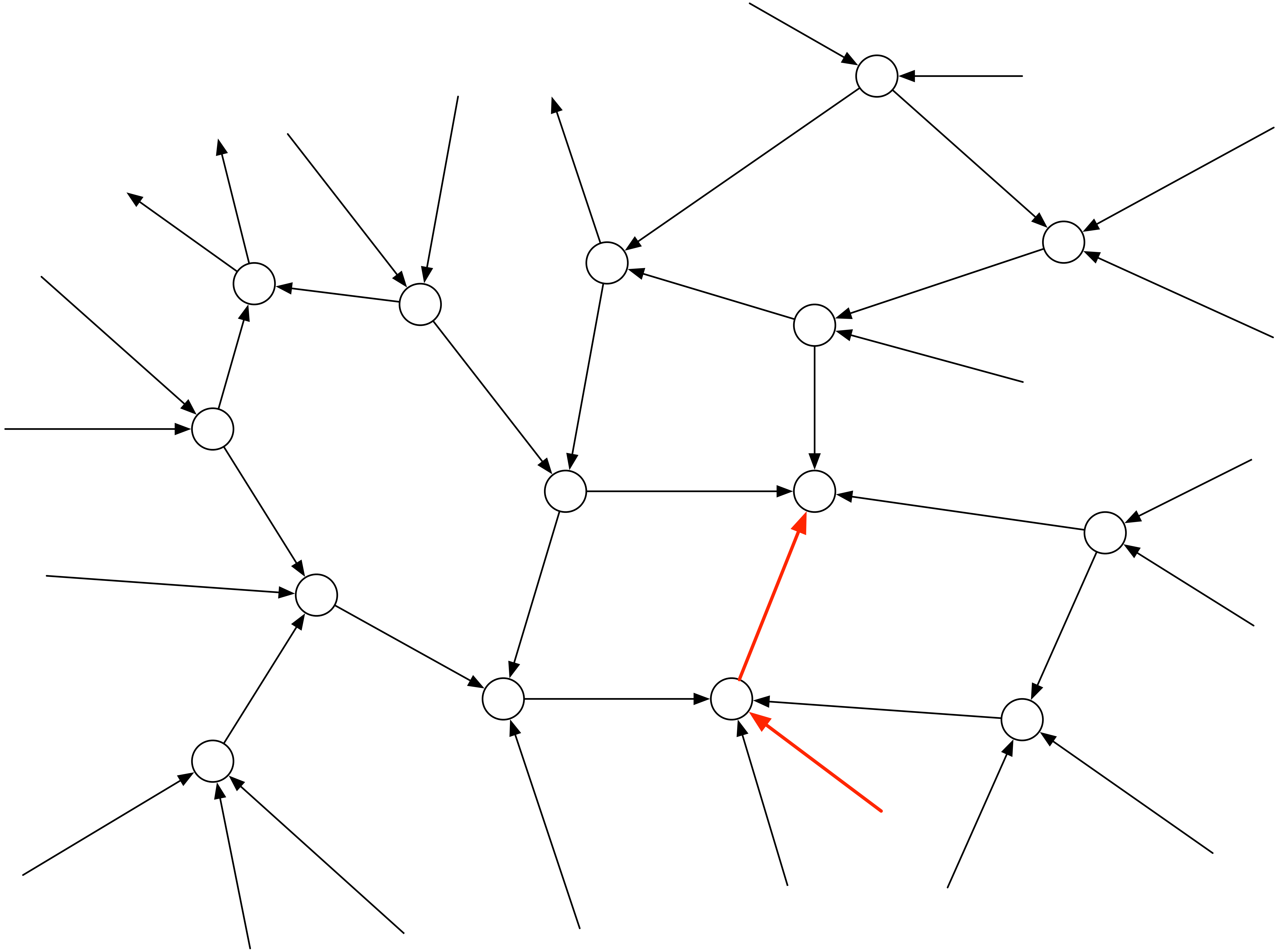
Fixing a solution



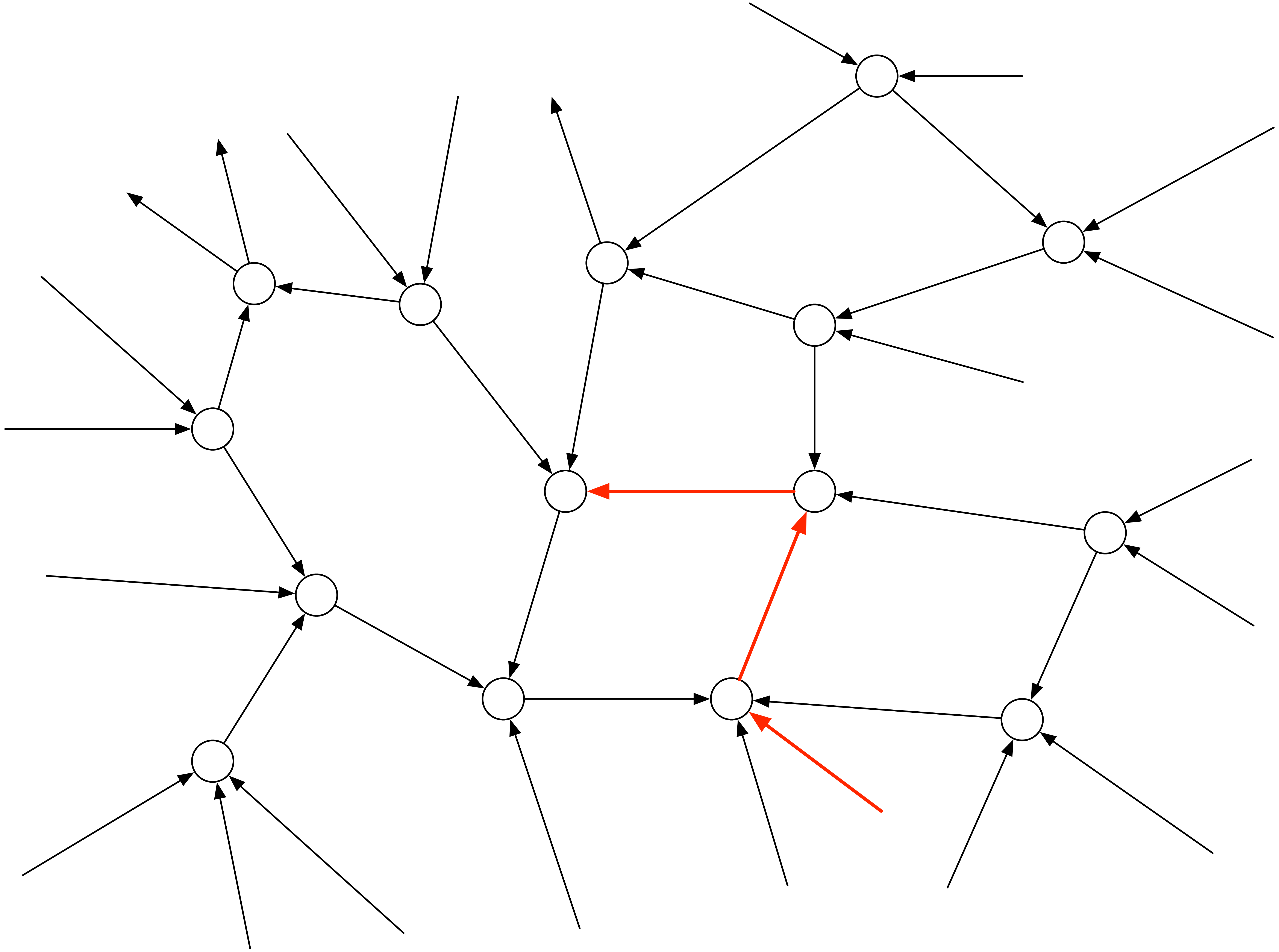
Fixing a solution



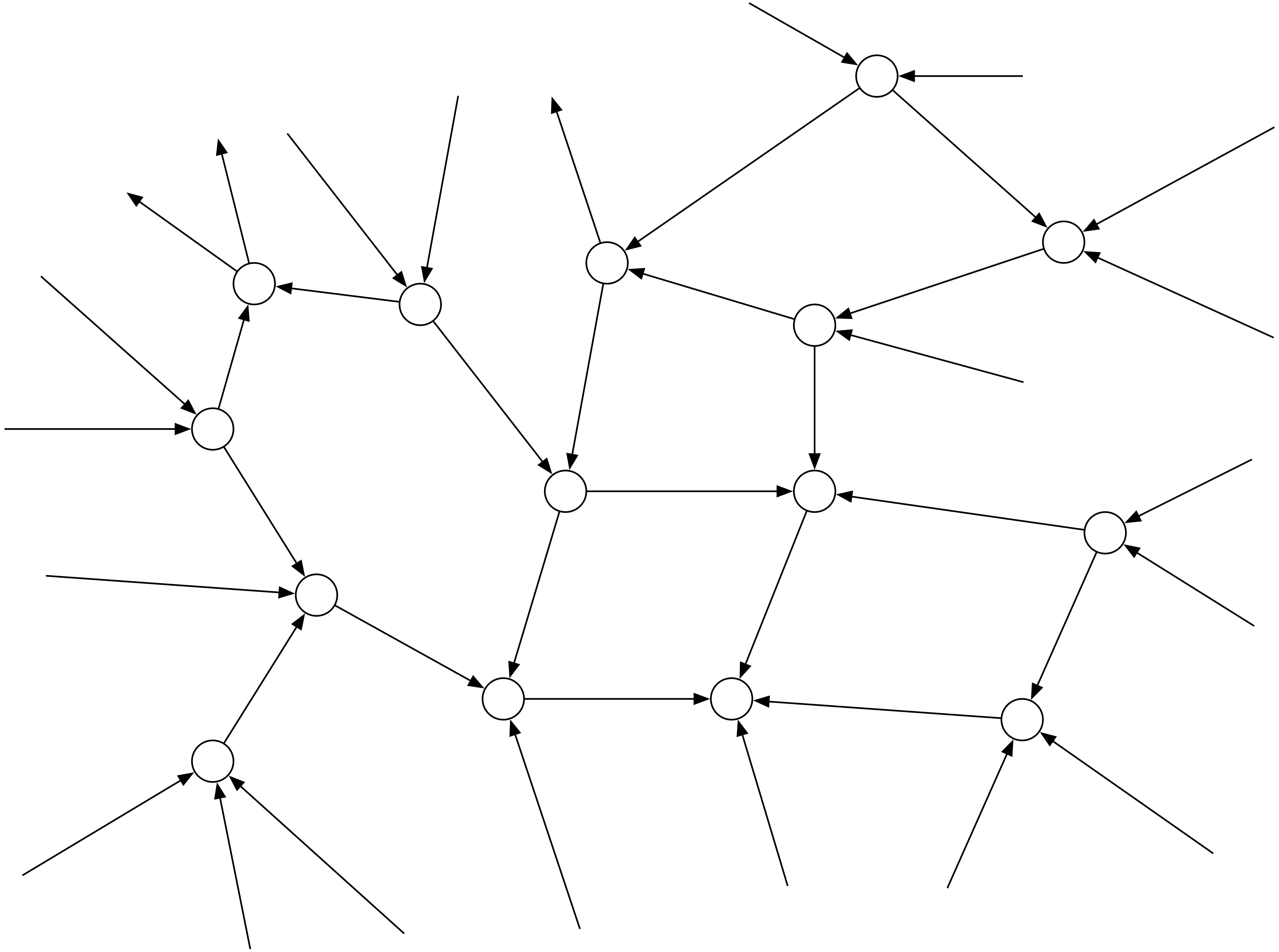
Fixing a solution



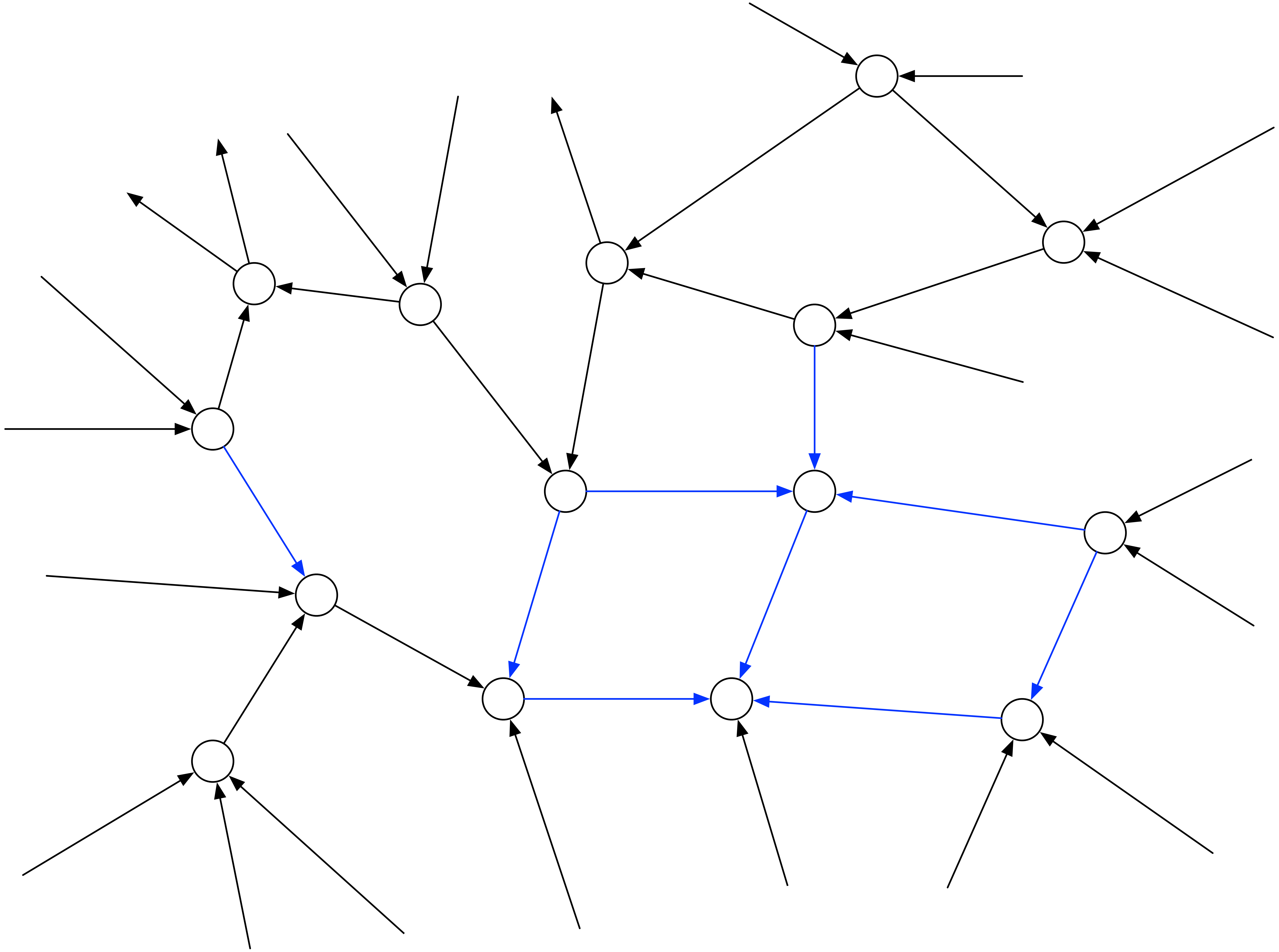
Fixing a solution



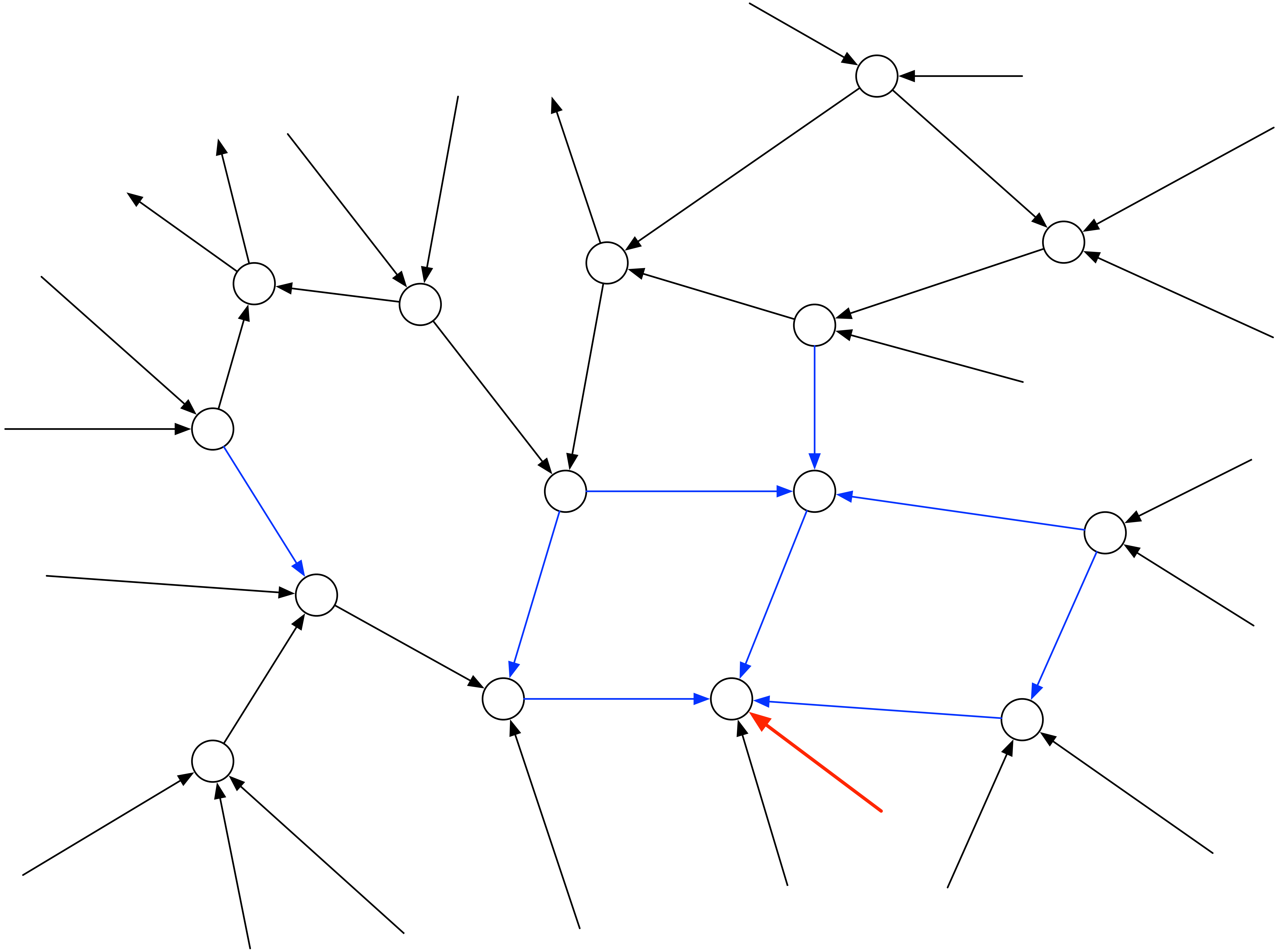
Token Dropping Game



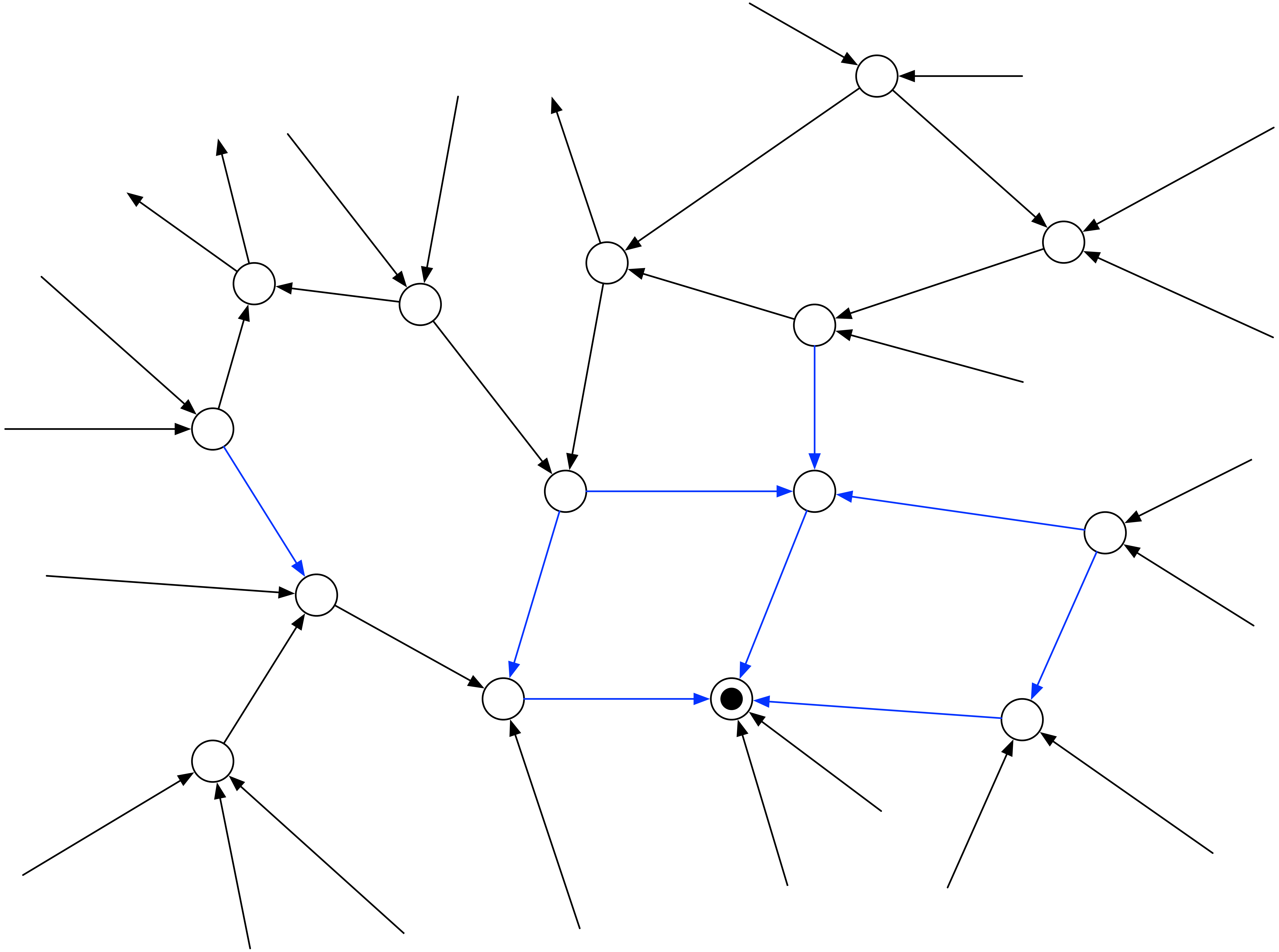
Token Dropping Game



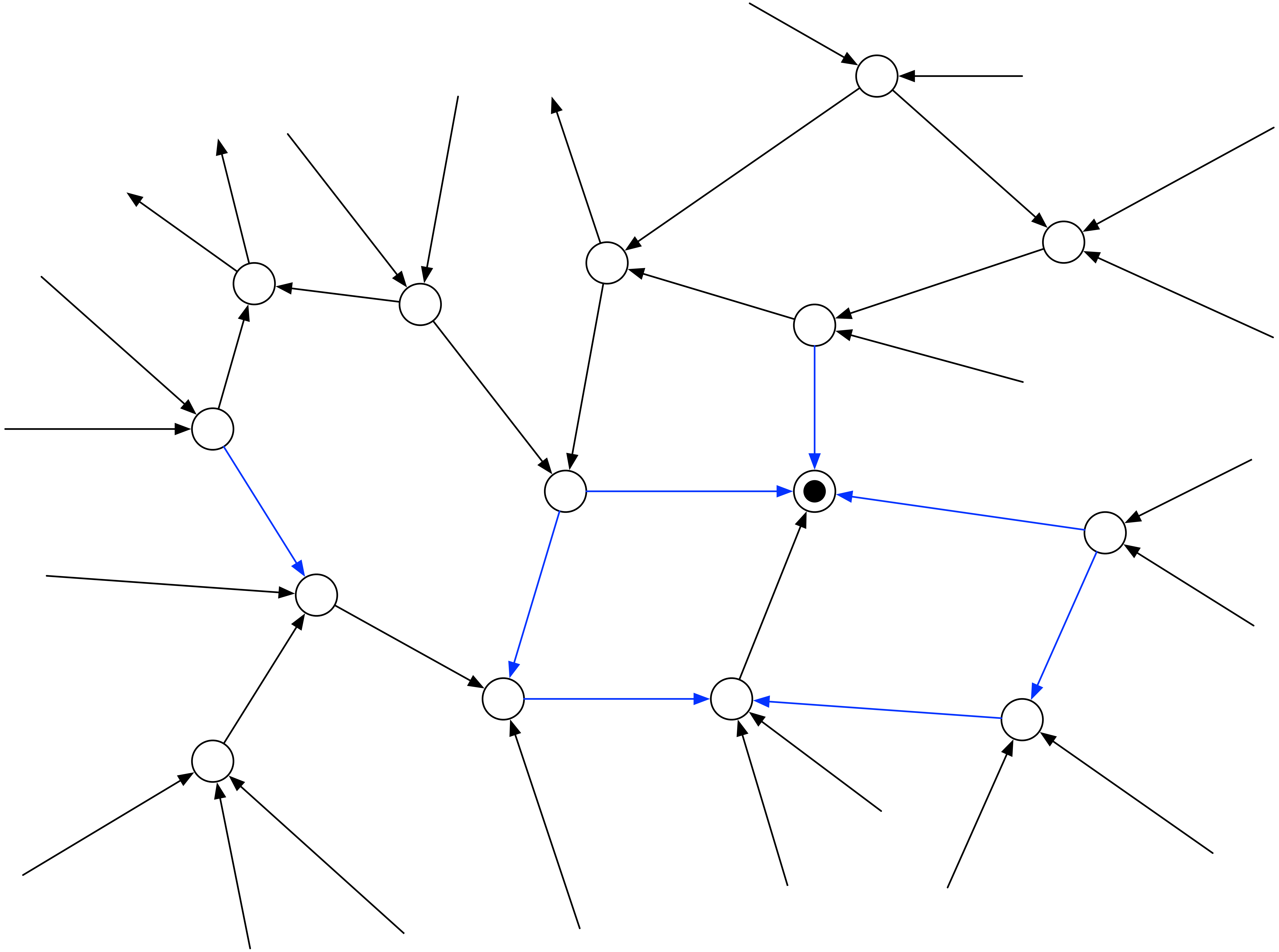
Token Dropping Game



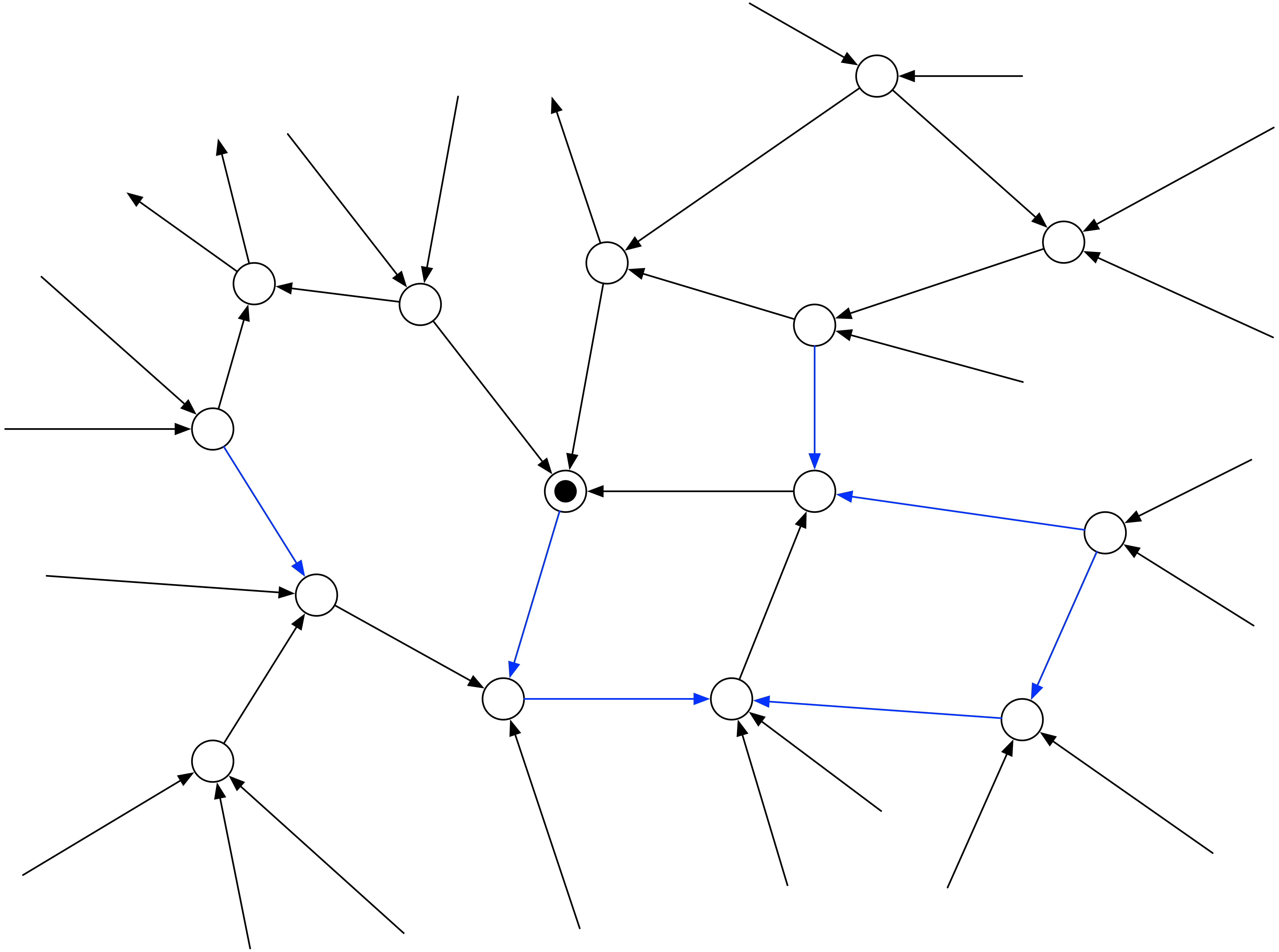
Token Dropping Game



Token Dropping Game

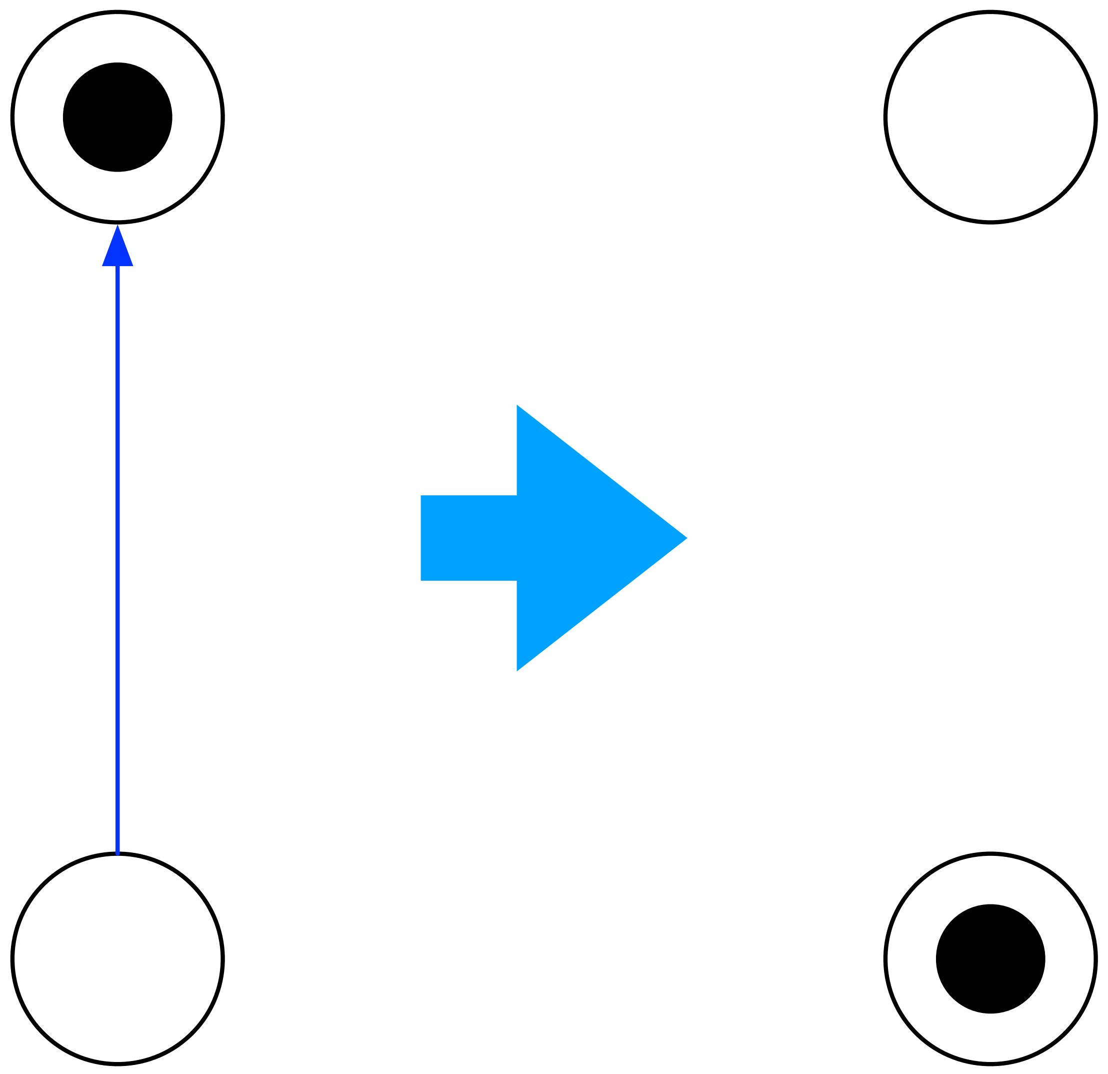


Token Dropping Game

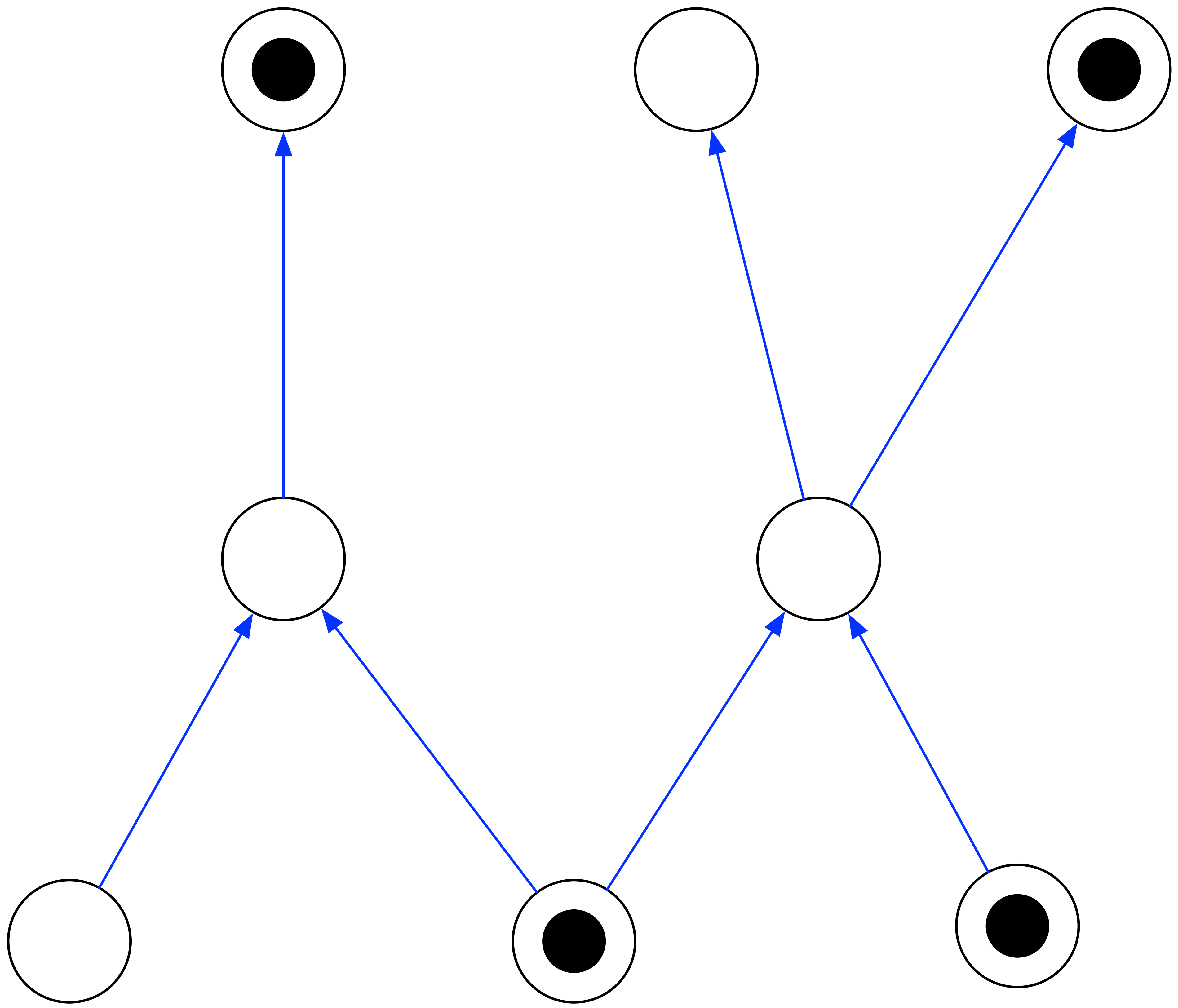


Token Dropping Game

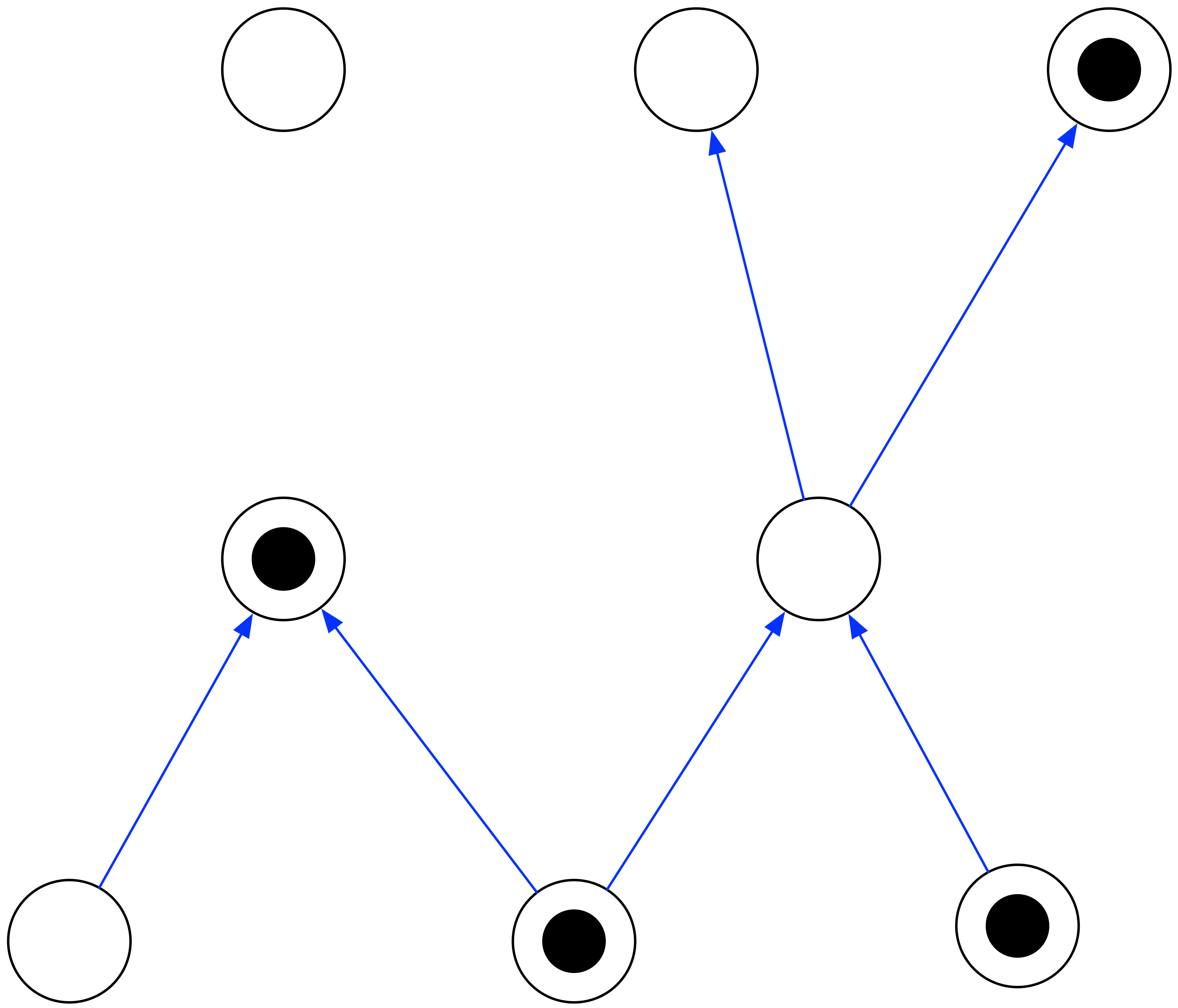
- We are given an **directed acyclic graph**
- Each **node** holds either **0 or 1 token**
- Tokens can be **moved** from **u** to **v** if and only if:
  - the edge **{u, v} exists**
  - the edge **{u, v} is oriented from v to u**
  - **u** is holding a **token**
  - **v** is **not** holding a **token**
- Once a token **passes** through an edge, the edge **disappears**
- We want to reach a **stable** solution



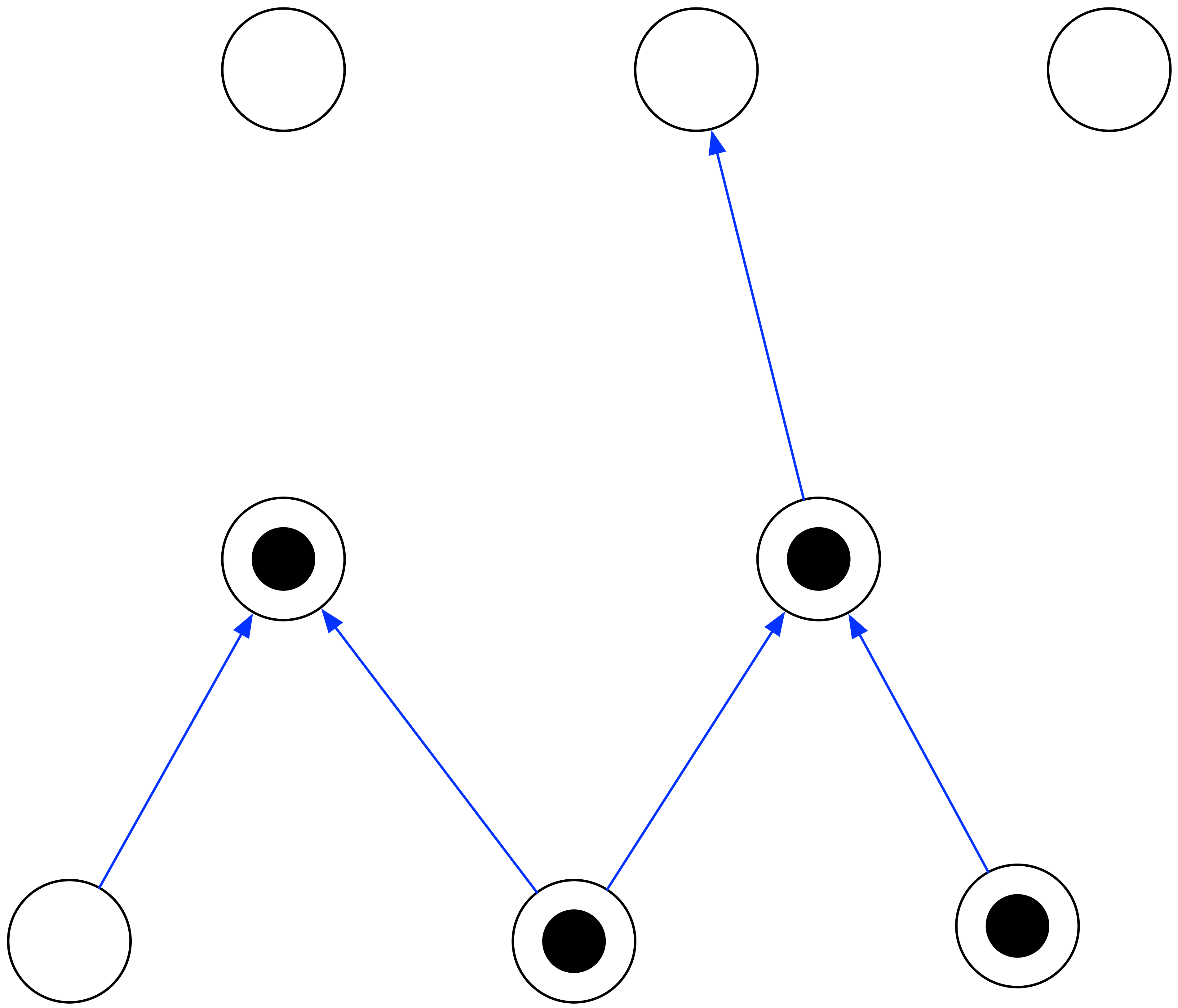
Token Dropping Game



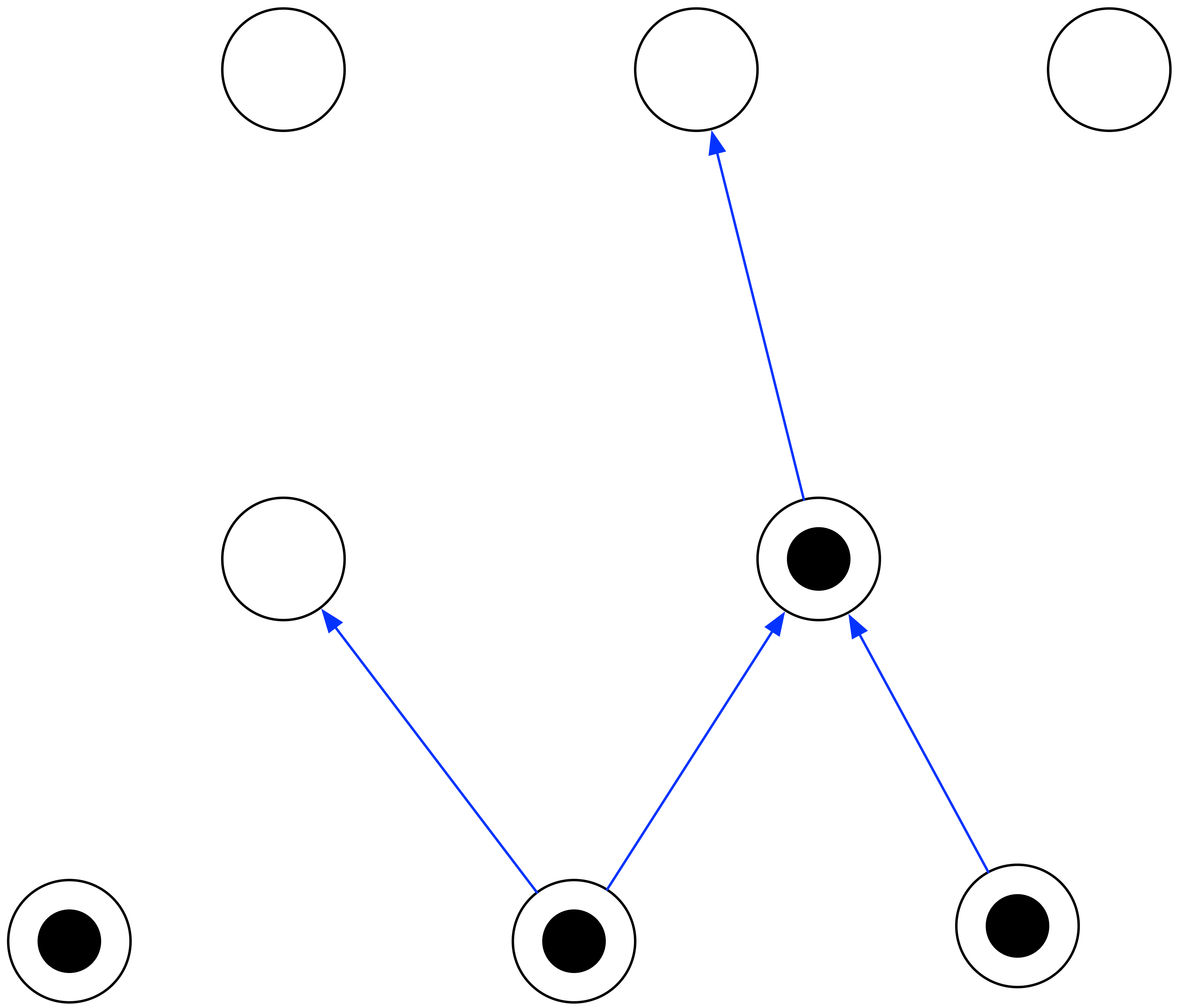
Token Dropping Game



Token Dropping Game



Token Dropping Game

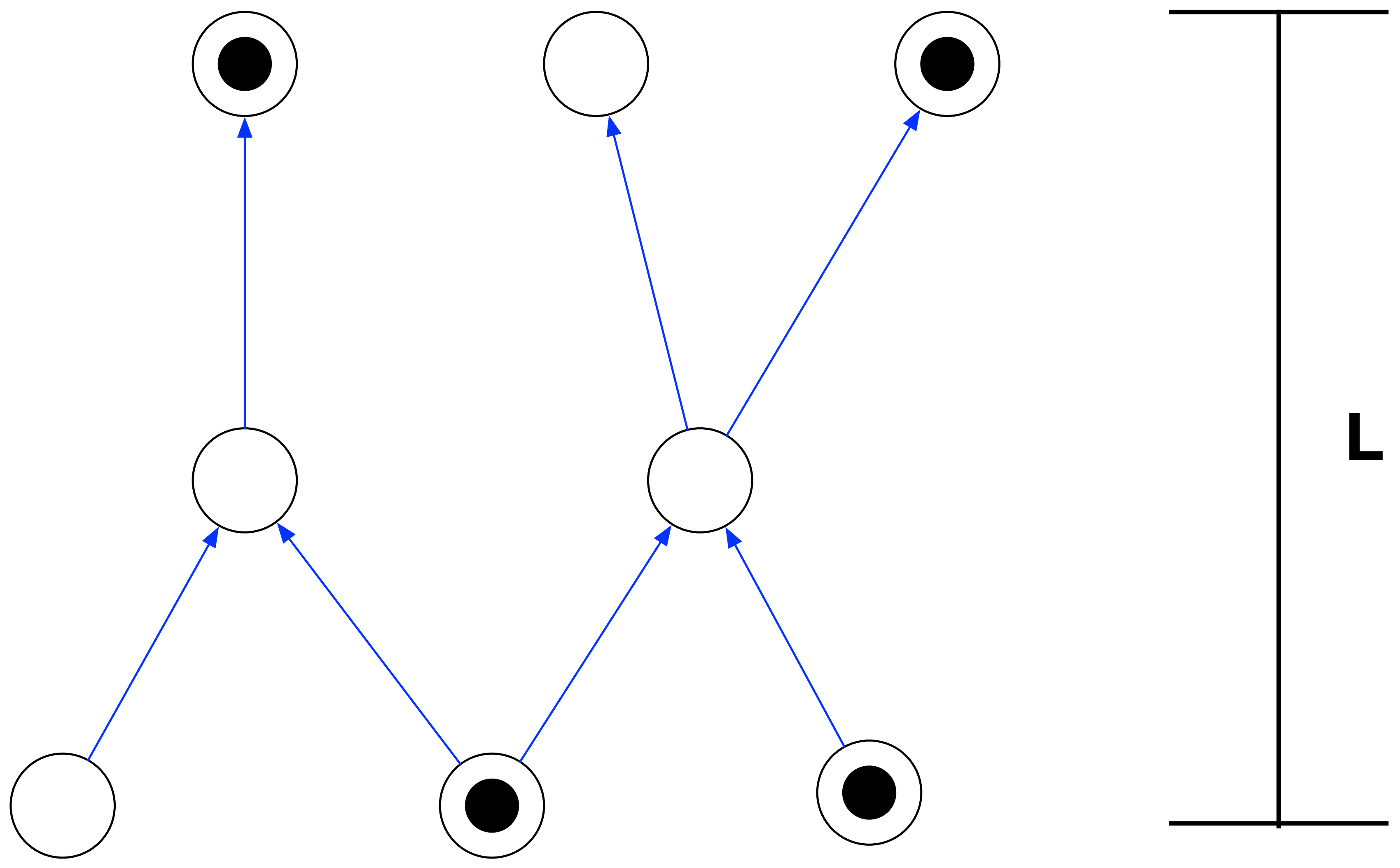


Token Dropping Game

Efficient Load-Balancing through Distributed Token Dropping

[Brandt, Keller, Rybicki, Suomela, Uitto 2021]

The Token Dropping Game can be solved in $O(L \cdot \Delta^2)$ rounds!



Semi Matching

Efficient Load-Balancing through Distributed Token Dropping

[Brandt, Keller, Rybicki, Suomela, Uitto 2021]

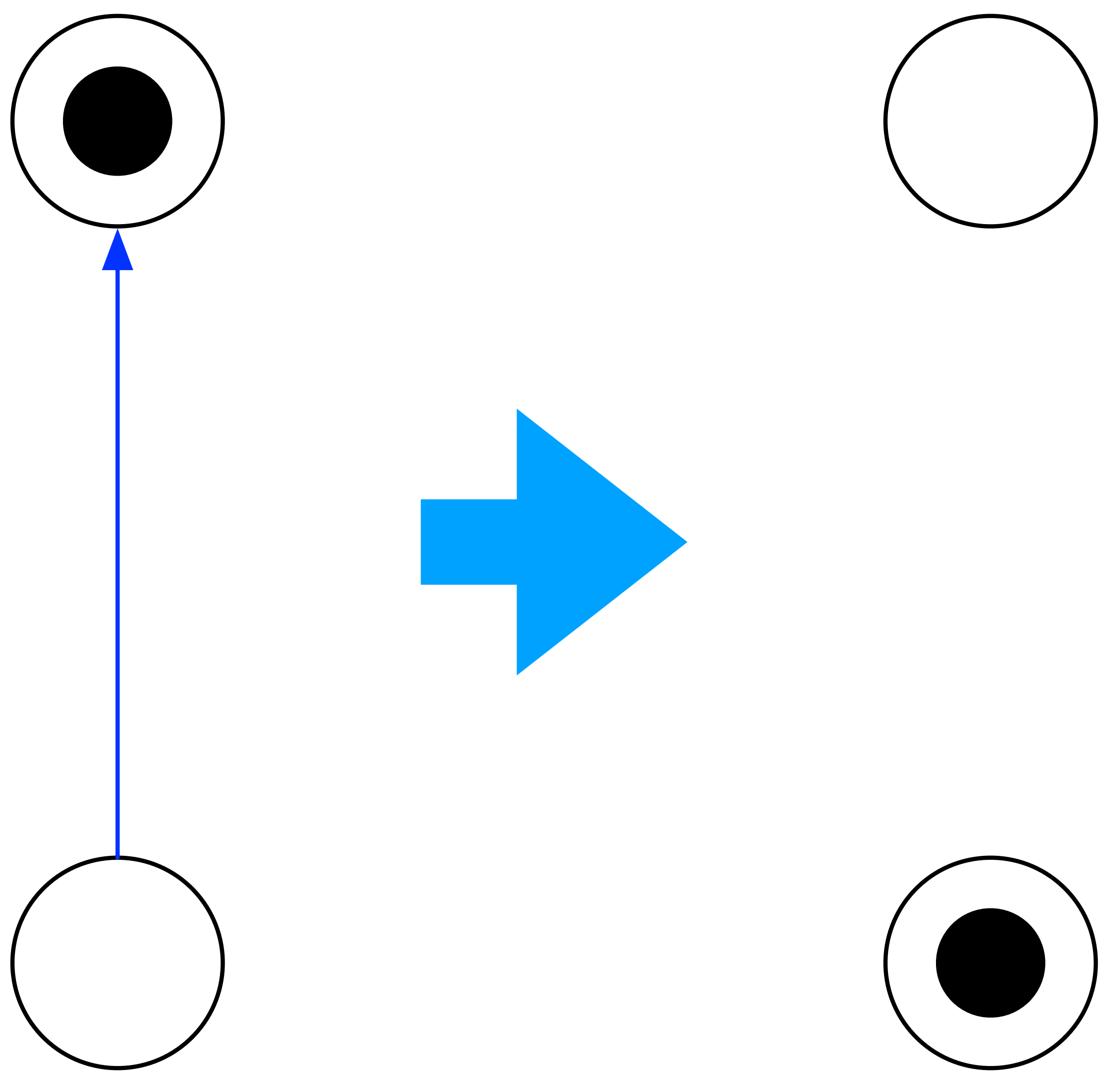
The Token Dropping Game can be solved in $O(L \cdot \Delta^2)$ rounds!

The Semi Matching problem can be solved by solving the Token Dropping Game for $O(\Delta)$ times (and $L = O(\Delta)$)

Semi Matching problem can be solved in $O(\Delta^4)$ rounds!

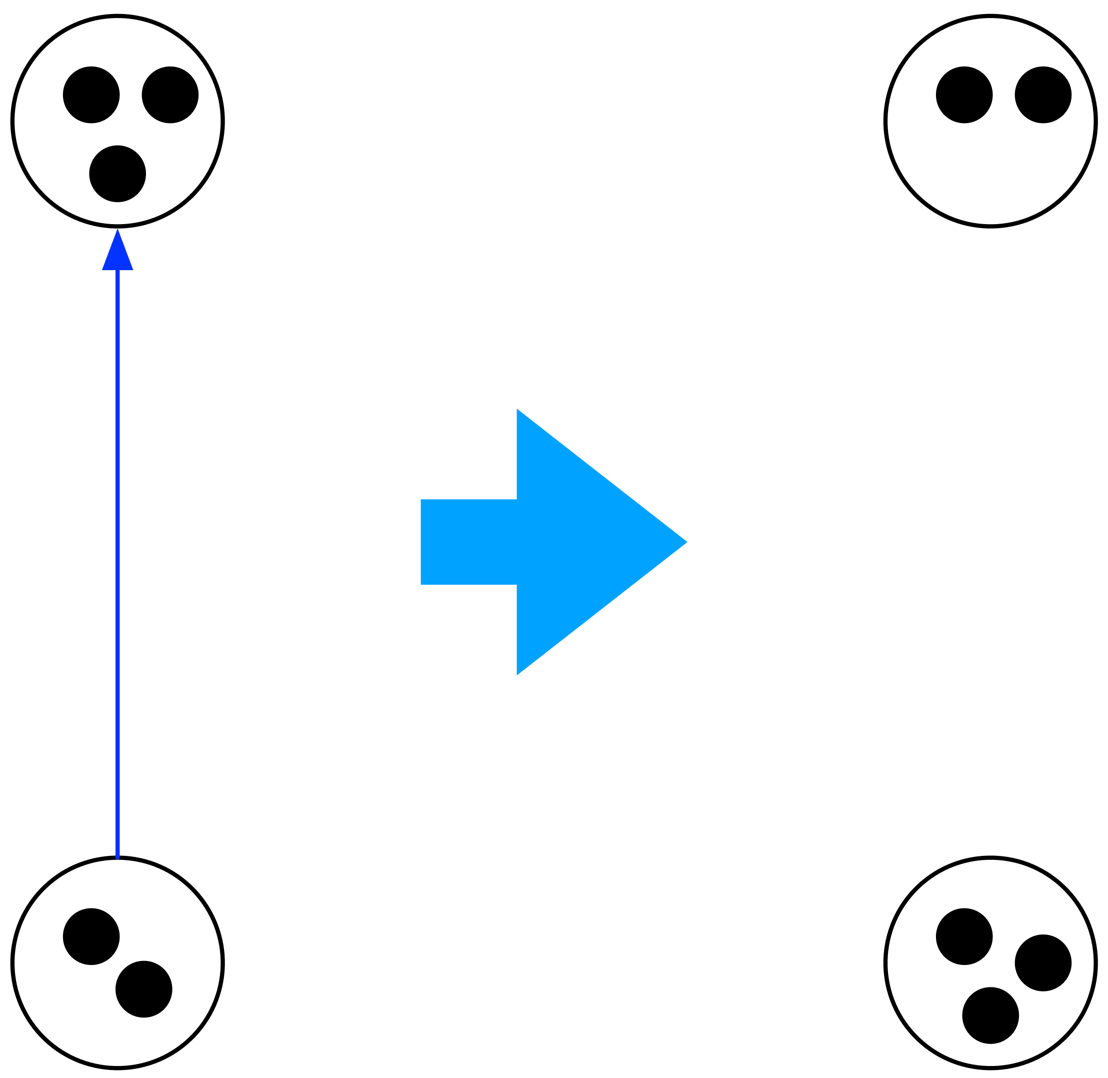
Token Dropping Game

- We are given an **directed acyclic graph**
- Each **node** holds either **0 or 1 token**
- A token can be **moved** from **u** to **v** if and only if:
  - the edge **{u, v} exists**
  - the edge **{u, v} is oriented from v to u**
  - **u** is holding a **token**
  - **v** is **not** holding a **token**
- Once a token **passes** through an edge, the edge **disappears**
- We want to reach a **stable** solution



Relaxed Token Dropping Game

- We are given an **directed acyclic graph**
- Each **node** holds **up to k tokens**
- A token **can** be **moved** from **u** to **v** if and only if:
  - the edge **{u, v} exists**
  - the edge **{u, v} is oriented from v to u**
  - **u** is holding **at least one token**
  - **v** is holding **at most k-1 tokens**, a token **must** move if **v** is holding **at most k/2** tokens
- Once a token **passes** through an edge, the edge **disappears**
- We want to reach a **stable** solution



Relaxed Token Dropping Game

Relaxed Token Dropping Game

- The Relaxed Token Dropping Game can be solved in $O(L \cdot \Delta^3/k^3)$ rounds!

Relaxed Token Dropping Game

- The Relaxed Token Dropping Game can be solved in $O(L \cdot \Delta^3/k^3)$ rounds!

- The Relaxed Semi Matching problem can be solved in $O(\Delta^5/k^5)$ rounds!

Relaxed Token Dropping Game

- The Relaxed Token Dropping Game can be solved in $O(L \cdot \Delta^3/k^3)$ rounds!
- The Relaxed Semi Matching problem can be solved in $O(\Delta^5/k^5)$ rounds!
- For $k = \dfrac{\Delta}{\log \Delta}$, this gives an $O(\log^5 \Delta)$ round algorithm!

Relaxed Token Dropping Game

- The Relaxed Token Dropping Game can be solved in $O(L \cdot \Delta^3/k^3)$ rounds!
- The Relaxed Semi Matching problem can be solved in $O(\Delta^5/k^5)$ rounds!
- For $k = \dfrac{\Delta}{\log \Delta}$, this gives an $O(\log^5 \Delta)$ round algorithm!
- With some $\operatorname{poly}\log \Delta$ overhead, we can solve $O(\Delta)$-edge coloring!

Relaxed Token Dropping Game

- The Relaxed Token Dropping Game can be solved in $O(L \cdot \Delta^3/k^3)$ rounds!

- The Relaxed Semi Matching problem can be solved in $O(\Delta^5/k^5)$ rounds!

- For $k = \dfrac{\Delta}{\log \Delta}$, this gives an $O(\log^5 \Delta)$ round algorithm!

- With some $\operatorname{poly}\log \Delta$ overhead, we can solve $O(\Delta)$-edge coloring!

- For $(2\Delta - 1)$-edge coloring, things are harder. E.g., the game is not even on a DAG!

Open questions: upper bounds

- We can solve $(2\Delta - 1)$-edge coloring in $O(\log^{12} \Delta)$ + $O(\log^* n)$
  - Can we improve the exponent? We know a faster algorithm, but only for $O(\Delta)$-edge coloring
- Can we solve vertex coloring in subpoly(Δ)?

Open questions: lower bounds

- Can we prove a non-trivial lower bound for solving $(2\Delta - 1)$-edge coloring?
  - Can we show that it cannot be solved in $o(\log \Delta) + O(\log^* n)$?

Thank you!